Numerical Simulations of the Ising Model on the Union Jack Lattice

Vincent Anthony Mellor

A thesis submitted for the degree of Doctor of Philosophy at

The University of Queensland in 18th November 2010

School of Mathematics and Physics

Declaration by author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my research higher degree candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the General Award Rules of The University of Queensland, immediately made available for research and study in accordance with the Copyright Act 1968.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material.

Statement of Contributions to Jointly Authored Works Contained in the Thesis

No jointly-authored works.

Statement of Contributions by Others to the Thesis as a Whole

Advice and suggestions throughout the degree from my supervisors Dr. Jon Links and Dr. Katrina Hibberd.

Statement of Parts of the Thesis Submitted to Qualify for the Award of

Another Degree

None

Published Works by the Author Incorporated into the Thesis

None

Additional Published Works by the Author Relevant to the Thesis but not Forming Part of it

None

Acknowledgements

Abstract

The Ising model is famous model for magnetic substances in Statistical Physics, and has been greatly studied in many forms. It was solved in one-dimension by Ernst Ising in 1925 and in two-dimensions without an external magnetic field by Lars Onsager in 1944. In this thesis we look at the anisotropic Ising model on the Union Jack lattice. This lattice is one of the few exactly solvable models which exhibits a re-entrant phase transition and so is of great interest.

Initially we cover the history of the Ising model and some possible applications outside the traditional magnetic substances. Background theory will be presented before briefly discussing the calculations for the one-dimensional and two-dimensional models. After this we will focus on the Union Jack lattice and specifically the work of Wu and Lin in their 1987 paper "Ising model on the Union Jack lattice as a free fermion model." [WL87]. Next we will develop a mean field prediction for the Union Jack lattice after first discussing mean field theory for other lattices. Finally we will present the results of numerical simulations. These simulations will be performed using a Monte Carlo method, specifically the Metropolis-Hastings algorithm, to simulate a Markov chain. Initially we calibrate our simulation program using the triangular lattice, before going on to run simulations for Ferromagnetic, Antiferromagnetic and Metamagnetic systems on the Union Jack lattice.

Keywords

Australian and New Zealand Standard Research Classifications (ANZSRC)

010506 Statistical Mechanics, Physical Combinatorics and Mathematical Aspects of Condensed Matter, 020304 Thermodynamics and Statistical Physics

Contents

List of Figures

Introduction

The Ising model is popular in statistical physics as a model for magnetic substances. The model can also be applied to systems outside of magnetic substances, such as diffusion of a gas and alloying in metals. Initially given to Ernst Ising as his PhD project by Wilhem Lenz in 1920, it is still the subject of many studies to this day. While Ising looked at the one-dimensional chain [Isi25], it was extended by Lars Onsager to the two-dimensional square lattice model without an external magnetic field in 1944 [Ons44]. The non-planar two-dimensional model, that is one that is with an external magnetic field, and models with higher dimensions, still remain unsolved except in rare cases [LCLG94].

In this thesis, we will investigate the Ising model on the Union Jack lattice and re-entrant phase transitions from the point of view of average magnetisation. In particular, we will investigate the results of Wu and Lin from their 1987 [WL87, WL88] and 1989 [WL89] papers. For this investigation we use three methods in the thesis. First we will perform a theoretical analysis of their results. Next we will develop a mean field simulation of the lattice and compare the results to the original exact solution. Finally we will perform numerical simulations on the various systems and compare these to the theoretical results.

1.1 Summary of the thesis

In Chapter 2 we will briefly revise some background information that will be required for the rest of the thesis. At the beginning of the chapter the thermodynamic laws will be presented and discussed. We will then move on to looking at some basic statistical mechanical concepts, such as partition functions and entropy. Once we have

a framework from the previous sections we will then move on to a discussion of phase transitions. Finally at the end of the chapter we will discuss the topic of magnetic substances from a chemistry point of view.

Using this background theory we will move on to a development of the theory of the Ising model in Chapter 3. We will look in detail at the development of the model, from magnetic sites, to the one-dimensional model and then extending to the two-dimensional square lattice model. At the end of the chapter the results for the triangular lattice are given. The triangular lattice is important as it will be used later for calibration of the simulation.

Having developed the theory for these lattices, we will move on in Chapter 4 to look at the Union Jack lattice model. Initially in the chapter, we will discuss the work of Vaks *et al.* [VLO66]. They use an older approach to calculate the critical temperatures of the system. They also discuss the re-entrant transition, with classification of the various phases. Then we will go on to look at the work of Wu and Lin [WL87, WL89], developing prediction functions for both sublattices. After this we will discuss their classification of phases at low temperatures. Once we have finished the presentation, we will then perform analysis on interesting systems and discuss the differences between the approaches.

In Chapter 5 we will examine the alternative approach of Mean Field theory. First we will review the theory on isotropic lattices giving results for the triangular lattice. From this we will develop equations for the Union Jack lattice. These will be in two forms, a partially uncoupled set of equations which will be dependent on only one sublattice magnetisation and a coupled set of equations. We will present simulations results using our mean field predictions and compare them to those from Wu and Lin [WL87, WL89]. We will also discuss the limitations of this approach and any disagreements with Wu and Lin's results. The simulation program will be discussed later in Appendix C.

At this point of the thesis we briefly move away from Ising models, to discuss the required theory for the next few chapters. Chapter 6 is concerned with the methods used for our numerical simulations. This section will be mainly concerned with Stochastic processes, in particular the Markov random walk. This will be simulated using a Monte Carlo method to reduce the necessary calculations. The Monte Carlo method we will be using is the Metropolis-Hasting algorithm. We will adapt this algorithm for

the simulation program. The actual program code will not be discussed here, but the interested reader may find such a discussion in Appendix B.

To calibrate our simulation program, in Chapter 7 we will run simulations on the triangular lattice. These simulations will range from the simplest isotropic ferromagnetic system, to the most general anisotropic antiferromagnetic system. We will compare our simulation results to the predictions of Baxter [Bax75] and Stephenson [Ste64] and comment on the correlation of results.

Our numerical simulation results on the Union Jack lattice will then be presented in Chapter 8. We will follow the structure of the theoretical analysis in Chapter 4, performing simulations on various systems of interest. These simulations will allow us to further investigate systems where there was a disagreement between the approaches on Vaks *et al.* [VLO66] and Wu and Lin [WL87, WL89]. Equally we will show how our simulation program allows the study of systems that are currently not able to be theoretically predicted.

To conclude the thesis, in Chapter 9 a review of the results from the theoretical, mean field theory and numerical approaches will be presented. We will then discuss how we can minimise the disagreement effects, such as additional conditions. A list of references will then follow.

1.2   History of the Ising model

In 1920, Wilhelm Lenz proposed a basic model of ferromagnetic substances to his then PhD student Ernst Ising. By 1925, Ising was submitting his dissertation [Isi25] which was the first exactly solved the one-dimensional case, and as such later was called the Lenz-Ising model. He discovered that there is no phase transition in this case. From this result he incorrectly extended it to say that there would be no phase transition in higher dimensional cases. Indeed during the early part of the twentieth century it was believed by some that the partition function would never show a phase transition, as the exponential function is analytic and as such the sum of these functions is also analytic. However, the logarithm of the partition function is not analytic near the critical temperature in the thermodynamic limit. The thermodynamic limit is reached as the number of particles in a system, N, approaches infinity. So in the infinite system the partition function is not analytic.

After Lenz and Ising proposed the model, in 1936 Peierls [Pei36] was able to explicitly show that a phase transition occurred in the two-dimensional model. He compared the high and low temperature limits and showed that at infinite temperatures all configurations have equal probability, while at high but not infinite temperatures there occurs clumping although the average magnetisation is still zero. At low temperatures the model resides in either a state with all the spins being positive or negative. Peierls questioned whether it was possible for the system to fluctuate between these states. In his work he managed to establish that the model defines super selection sectors. That is, domains that are not connected by finite fluctuations.

While greats of physics such as Heisenberg (1928) [Hei28], Kramers and Wannier (1941) [KW41a, KW41b] use the model to examine ferromagnetism and properties such as the Curie temperature (the critical temperature of substances), it was almost 20 years before Lars Onsager had presented a solution to the two-dimensional model with no external field [Ons44]. His paper and solution, although algebraically complex, was a large step in understanding the model further. While other solutions have been developed that are less algebraically complex, other than in special cases, it has only been solved for a zero external magnetic field by Schultz *et al.* [SML64]. This solution will be presented later in Chapter 3. The derivation of the solution was published by Yang in [Yan52], but Onsager presented the result on the 28 February 1942 at a meeting of the New York Academy of Science.

The extension to three dimensions is more complex. In the paper of Schultz *et al* [SML64], the method is limited by the transformation of the system into fermion annihilation and creation operators. In 1972, Richard Feynman [Fey72] said of the three-dimensional Ising model, "the exact solution for three dimensions has not yet been found." In 2000, Sorin Istrail [Ist00], by extending Francisco Barahona's work [Bar82], showed that the solution to the three-dimensional problem as well as the two-dimensional non-planar model is NP-complete. He used a method called computational intractability, which allows one to discover if a problem can be solved in a feasible timeframe[1]. As there are about 6000 such problems, and they are all mathematically equivalent, a solution to one would be a solution to all and this is an infeasible result. While this result does not remove the possibility of a solution (the P versus NP question is still open), it does mean that the solution is not possible using the methods

---

[1]Less that the lifetime of a human

known today. However, the three-dimensional problem, as with the non-planar two-dimensional model, can be solved approximately numerically and in some special cases [LS74, SSSEA06, Suz76].

In this section, I have presented a brief history showing the major milestones in the development of the model in various dimensions, for the interested reader there are many more detailed discussions of the history, such as [Bru67] or [PB94].

1.3   Applications of the Ising model

The Ising model has been applied to a great variety of other physical systems such as absorption of gases on to solid surfaces, order-disorder transitions in alloys, concentrated solutions of liquids, the helix-coil transition polypeptides and the absorption of oxygen by haemoglobin. [VS95, Man88, BD06]

An example, taken from [Man88], of an order-disorder transition in alloys is the binary alloy of Copper and Zinc, brass. Brass has many forms depending on the atomic percentage (that is, the number of atoms) of these metals and type of lattice. For example α-brass generally have below 35 atomic percent Zinc, a single phase and is made of a face-centred lattice. At compositions in a narrow range around 50 atomic percent Copper, 50 atomic percent Zinc, the atoms occupy the sites of a body-centred cubic lattice (as show in Figure 1.1) forming β-brass. A body-centred cubic lattice consists of two interlocking simple cubic sublattices, where one site of one lattice is centred in the centre of a cube of the other lattice. This system undergoes a phase transition at the critical temperature, T_c, of about 740 K. The distribution of atoms on these sites is disordered above a temperature T_c. Below T_c there is ordering with atoms of each kind preferentially distributed on one of the two simple cubic sublattices of the body-centred cubic sublattices of the body-centred cubic lattice (β' phase).

1.4   Related works

In this research we use a simulation program which uses a Metropolis-Hastings algorithm. This algorithm produces local changes in the system, which means the configuration is changed only in a small neighbourhood of each update. While there are advantages to this method, near the critical point there is a slowing down phenomenon.
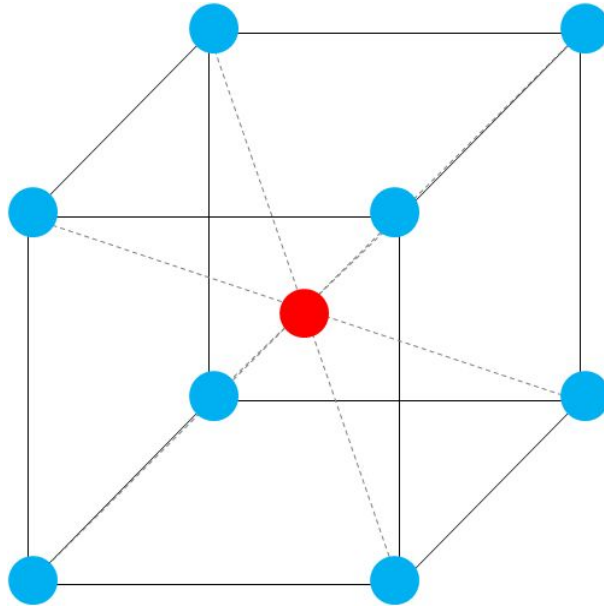
Figure 1.1: Simplified representation of the body centred cubic of the β' phase of brass where the red atom is Copper and the blue atoms are Zinc.

This means that with an increase in lattice size the computation time increases rapidly so the numerical study has been restricted to small systems. To study larger systems it has been suggested by Blöete *et al.* in [BHL02] that cluster Monte Carlo methods can be used. In these methods, instead of flipping each individual site per iteration, clusters of sites are now flipped. This allows more efficient algorithms, such as those from Swendsen and Wang [SW87], Baillie and Coddington [BC91] and Wolff [Wol89]. The Wolff single-cluster algorithm [Wol89] stands out because of its simplicity with only one cluster being formed and flipped at a time. However, as cluster analysis is not as easy to generalise as the local interaction methods, its applicability range is limited. One possible pitfall of cluster algorithms is in the situation where the clusters tend to occupy practically the whole system, resulting in only trivial changes of the spin configuration, and in a limited efficiency.

The work of Wu and Lin has be extended by Strečka to mixed spin Ising models, in his papers [Str06] and [SvD06]. Mixed spin models are systems where the spins have different magnitudes. These mixed spin models have a much richer critical behaviour compared to the single spin models considered in this thesis. They are also useful as they describe the simplest ferrimagnets (see Section 2.5) and so have a wide range of practical applicability. In his solo paper [Str06], Strečka investigates the spin-1/2 and

spin-3/2 Ising model on the Union Jack lattice. In the article he presents a formulation of the system and its equivalency to the eight-vertex model, and then performs numerical simulations to investigate the critical points. In [SvD06], he and his co-authors investigate the mixed spin-$(1/2, S)$ Ising model, and compare the integer versus half-odd-integer spin-S case. In the paper they extend Strečka's previous work and examine how critical exponents may depend on the quantum spin number. For example, the mixed spin-$(1/2, 1)$ model exhibits quite different variations of the critical exponents when compared to the $(1/2, 3/2)$ version.

Theory Revision

In this chapter we will present a brief review of some of the background theory that will be used in later sections. First we will look at the topic of thermodynamics, discussing both the motivation and the laws of thermodynamics. Then we will move on to look at some important topics in statistical physics like the partition function. Finally we will present an overview of the chemistry of magnetic substances.

The topics in this section are commonly known results that form a basis for a great amount of work. As such discussions on these topics in statistical physics can be found in [Man88, PB94] and for magnetochemistry [Ear68, Orc03, Sha92, SAL90], along with many other books on both subjects.

2.1 Motivation for thermodynamics

A macroscopic system has many unmeasurable quantities that affect the overall behaviour of the system. Thermodynamics concerns itself with the relation between a small number of variables that are sufficient to describe the bulk behaviour of the system in question. In the case of a magnetic solid the variables are the magnetic field $\mathbf{B}$, the magnetisation $\mathbf{M}$, and the temperature T. If the thermodynamic variables are independent of time, the system is said to be in *steady state*. If moreover there are no macroscopic currents in the system, such as a flow of heat or particles through the material, the system is in *equilibrium*.

A thermodynamical transformation or process is any change in the state variables of the system. A *spontaneous* process is one that takes place without any change in the external constraints on the system, due simply to the internal dynamics of the system. An *adiabatic* process is one in which no heat is exchanged between the system and its

surroundings. A process is *isothermal* if the temperature is held fixed. A *reversible* process follows a path in thermodynamic space that can be exactly reversed. If this is not possible, the process is *irreversible.*

2.2 Laws of thermodynamics

The laws of thermodynamics are relations that have been found to describe the properties of a system when going through a thermodynamic process. These laws govern properties such as the direction that a process progresses in (which is stated by the second law). They also help to relate thermodynamics to mechanical ideas, such as the conservation of energy.

Zeroth law

Many quantities like pressure and volume have direct meaning in mechanical systems. However some basic quantities of statistical mechanics are quite foreign to mechanical systems. One of these concepts is that of temperature. Originally temperature relates to the sensations of hot and cold of a particular system. A remarkable feature of temperature is its tendency to equilibrium. One can see this when taking a soda can out of the fridge. When it is initially out of the fridge it is cooler than the surrounding room, but it slowly warms up to be in equilibrium, that is the same temperature, with the surroundings. This type of equilibrium is often referred to as thermal equilibrium.

The zeroth law is used to assign a measurable value to this concept of temperature. Formally the law can be stated as:

> If a system A is in equilibrium with systems B and C, then B is in equilibrium with C.

That is if two systems are in the same temperature as a third common system then they are the same temperature as each other. In general thermometers are systems where a property that depends on its degree of hotness can be measured equally, such as the electric resistance of a platinum wire. Once calibrated with some known temperatures, such as the ice and steam points of water, and interpolating linearly for other temperatures, one can use this system to measure the temperature of other

systems. Of course temperature scales depend on the particular thermometer used. This arbitrariness can be removed by developing an *absolute temperature scale* using the second law of thermodynamics (2.2)

First law

The first law of thermodynamics restates the law of conservation of energy. However, it also partitions the change in energy of a system into two pieces, heat and work:

$$dE = \bar{d}Q - \bar{d}W. \qquad (2.1)$$

In (2.1) dE is the change in internal energy of the system, $\bar{d}Q$ the amount of heat *added* to the system, and $\bar{d}W$ the amount of work *done* by the system during an infinitesimal process. Apart from the partitioning of energy into two parts, the formula distinguishes between the infinitesimals dE and $\bar{d}Q$, $\bar{d}W$. The difference between the two measurable quantities $\bar{d}Q$ and $\bar{d}W$ is found to be the same for any process in which the system evolves between two given states, independently of the path. This indicates that dE is an exact differential or, equivalently, that the internal energy is a function based on the state of the system. The same is not true of the differentials $\bar{d}Q$ and $\bar{d}W$, hence the difference in notation.

Consider a system whose state can be specified by the values of a set of the sate variables x_j (for example the magnetisation, et cetera) and the temperature. Thermodynamics exploits an analogy with mechanics and so, for the work done during an infinitesimal process,

$$dW = -\sum_j X_j dx_j$$

where the X_j's can be though of as generalised forces and the x_j's as generalised displacements.

Second law

The second law of thermodynamics introduces the *entropy*, S, as a state variable and states that for an infinitesimal reversible process at temperature T, the heat given to the system is

$$dQ_{rev} = TdS \qquad (2.2)$$

while for an irreversible process

$$dQ_{irrev} \leq TdS$$

If the only interest is in thermodynamic equilibrium states, (2.2) can be used and then the entropy S can be treated as the generalised displacement that is coupled to the force T. The above formulation of the second law is due to Gibbs.

Two equivalent statements of the second law of thermodynamics are: The *Kelvin* version

> There exists no thermodynamic process whose sole effect is to extract a quantity of heat from a system and convert it entirely to work.

and the equivalent statement of *Clausius*

> No process exists in which the sole effect is that heat flows from a reservoir at a given temperature to a reservoir at a higher temperature.

Third law

The third law of thermodynamics is of a more limited use, with its main applications being in chemistry and low temperature physics. It originated in Nernst's (1906) study of chemical reactions, and is sometimes called Nernst's Theorem. While it is of some interest in the report and it will briefly discussed, a more detailed treatment can be found in [Wil61, tH66].

The third law of thermodynamics is concerned with the entropy of a system as the temperature approaches absolute zero. Stated simply it says that at absolute zero all bodies will have the same entropy. This means that at absolute zero a body will be in the only possible energy state, so would possess a definite energy (called the *zero-point energy* and would have zero entropy in this state. This value is left undetermined by purely thermodynamic relations such as the previous laws, which only give entropy differences. However a postulate related to the second law, although independent from it, is that it is impossible to cool a body to absolute zero by any finite process. Although you can get as close as you desire, you can not actually reach the absolute zero.

An alternative version of the third law is given by Gilbert N. Lewis and Merle Randall in 1923

> If the entropy of each element in some (perfect) crystalline state be taken as zero at the absolute zero of temperature, every substance has a finite positive entropy; but at the absolute zero of temperature the entropy may become zero, and does so become in the case of perfect crystalline substances.

Of course there are substances where at absolute zero there exists a residual entropy, such as carbon monoxide, but this is because there exist more than one ground state with the same zero point energy.

2.3 Partition functions

The partition function (Z) [Man88] is a quantity in Statistical Mechanics that encodes the statistical properties of a system in thermodynamic equilibrium. It is a function of temperature and other parameters, such as the volume enclosing a gas. Using the partition function of a system one can derive quantities such as the average energy, free energy, entropy or pressure from the function itself and its derivatives.

There are many forms of the partition function depending on the type of statistical ensemble used. The two major versions are the *Canonical partition function* and the *Grand canonical partition function* [PB94]. The canonical partition function applies to a canonical ensemble, that is a system that is allowed to exchange heat with the environment at a fixed temperature, volume and number of particles. The grand canonical partition function applies to the grand canonical ensemble, where a system is allowed to exchange heat *and particles* with the environment, at a fixed temperature, volume and chemical potential. Of course for other circumstances one can define other partition functions. In this report we will be using the canonical partition function exclusively, so we will briefly develop this here, and refer to it from now on as the partition function.

2.3.1 Definition

Assume we are looking at a thermodynamically large system that is in constant thermal contact with its surroundings, which have temperature T, and the system's volume and number of particles have been fixed, in other words a system in the canonical ensemble.

If we label the microstates (exact states) that the system can take with s ($s = 1, 2, \ldots$) and the energy of these microstates as E_s then the partition function will be

$$Z = \sum_s e^{-\beta E_s} \tag{2.3}$$

where the "inverse temperature" β is defined as

$$\beta \equiv \frac{1}{k_B T}.$$

In quantum mechanics, the partition function can be written more formally as the trace over the state space

$$Z = \mathrm{Tr}\left(e^{-\beta H}\right)$$

where H is the quantum Hamiltonian operator.

2.3.2   Using the partition function

It may not be obvious why the partition function is of such importance to Statistical Mechanics from the definition above. The ability to calculated the microstates of a system using a particular model, and from those to calculate a function that can be used to calculate the thermodynamical properties of the system is a very powerful tool. The partition function has a very important statistical meaning which allows this relation to the thermodynamical properties. The probability P_s that a system is in state s is

$$P_s = \frac{1}{Z} e^{-\beta E_s}. \tag{2.4}$$

$e^{-\beta E_s}$ is the well known *Boltzmann factor* or *Boltzmann weight* [Man88]. As we can see the partition function is playing the role of a normalising constant for this, which ensures that the probabilities add up to one:

$$\sum_s P_s = \frac{1}{Z} \sum_s e^{-\beta E_s} = \frac{1}{Z} Z = 1.$$

It is for this reason Z is called the "partition function"; it weights the different microstates based on their individual energies. The reason for the partition function to be called Z is because it is the first letter of the German word for sum over states,

"Zustandssumme".

We can go on to use the partition function to calculate the average energy of the system. This is simply the expected value of the energy, which of course is the total of the energies of the microstates weighted by their relative probabilities.

$$\langle E \rangle = \sum_s E_s P_s = \frac{1}{Z} \sum_s E_s e^{-\beta E_s} = -\frac{\partial \ln Z}{\partial \beta} \tag{2.5}$$

or equivalently,

$$\langle E \rangle = k_B T^2 \frac{\partial \ln Z}{\partial T}.$$

2.3.3   Entropy in statistical physics

The idea of entropy was introduced above in the second law of thermodynamics, and it was even defined as a state variable, but what is entropy? The entropy of a system is a measure of the disorder of the system and is related to the energy of the system. Each microstate, which are the states at the microscopic level, have statistical weight related to the energy of the system. The entropy of these microstates can be defined from these statistical weights by using the following formula,

$$S = k_B \ln \Omega(E) \tag{2.6}$$

where $\Omega(E)$ is the number of the microstates with energy E. The process in which the system is heated up without any work is reversible. Hence

$$dE = \bar{d}Q_{rev} = TdS.$$

Therefore, by rearranging this formula

$$\frac{\partial S}{\partial E} = \frac{1}{T} \tag{2.7}$$

where T is the temperature of the system. This shows that the entropy will increase slower as the temperature gets higher, but the entropy will never decrease as the temperature increases.

As we have seen above, one can calculate many quantities with the partition func-

tion, and the entropy of the system is one of those quantities.

$$S \equiv -k_B \left(\ln Z + \beta \left\langle E \right\rangle \right) = \frac{\partial}{\partial T} \left(k_B T \ln Z \right)$$

2.4   Phase transitions

As we are studying the physics of phase transitions we should, at this point, define what we mean by a phase transition. To do this first let us define what we mean by a phase.

A phase is a homogeneous part of a system bounded by surfaces across which the properties change discontinuously. In the most general situation each phase will contain several components, that is, it will contain several different species of molecules or ions. For example a gaseous phase might consist of a mixture of gases. To specify the properties of a phase we must then specify the concentrations of the various components in it. Then we have the possibility of transfer of matter between different phases. This matter has then gone through a phase transition.

A first order phase transition is normally accompanied by the absorption or liberation of latent heat, while a second order phase transition has no latent heat. An example of a first order phase transition is between water and steam (gaseous water), and an example of a second order phase transition that exhibited by is the two=dimensional Ising model on a square lattice (magnetisation). There is a critical temperature and energy at which the properties of the two phases of the transition are identical. A phase transition is easily seen on a graph of the spontaneous magnetism versus the temperature of the system as a discontinuity of the data, or a point where the temperature remains constant while the energy input increases. Further information on phase transitions can be found in [Man88].

2.5   Magnetic substances

Substances are made up of smaller particles that cannot be divided without loss of the description of the substance, that is, if these particles were divided, the parts would not exhibit all the properties of the substance. Depending on the bonding of the substance, these particles may contain smaller particles called ions or it may be a

molecule of the substance. Ions are of a similar configuration as atoms, with electrons orbiting around a nucleus, but have a different electron configuration. If the ion has more electrons than the base atom, then the ion will have a negative charge, and with less electrons will have a positive charge. Molecules are collections of atoms that share their electrons among each other by using overlapping orbits, which is called a covalent bond. As it is sufficient to describe the properties of the molecules or ions, because the substance is then in its most basic form, we shall concentrate on the description of these particles initially. Then we shall look at a crystal lattice structure of atoms or ions to investigate the possible properties that might be found as a result of the arrangement of the particles. It is sufficient to only describe a crystal lattice with ionic bonding as it can be generalised for any other structure with any type of bonding.

For a molecule or ion to be paramagnetic [Ear68, Orc03], that is, attracted by a magnetic field, it must contain one or more unpaired electrons. Substances that are repelled by magnetic fields are called diamagnetic and contain no unpaired electrons. The magnetic moment of a paramagnetic species increases with the increase in the number of unpaired electrons. Liquid oxygen is easily shown to be paramagnetic by pouring it between the poles of a strong magnet, when it is attracted by the field and fills the gap between the poles.

Substances in which the paramagnetic species are separated from one another by several diamagnetic species are said to be magnetically dilute [Sha92]. When the paramagnetic species are very close together (as in metals) or are separated only by an atom or monatomic ion that can transmit magnetic interactions they may interact with one another. This interaction may lead to ferromagnetism (in which large domains of magnetic dipoles are aligned in the same direction) or antiferromagnetism (in which neighbouring magnetic dipoles are aligned in opposite directions) [SAL90]. *Ferromagnetism* leads to greatly enhanced paramagnetism as in iron at temperatures up to 1041 K (the Curie temperature), above which thermal energy overcomes the alignment and normal paramagnetic behaviour occurs. *Antiferromagnetism* occurs below a certain temperature called the Néel temperature; as less thermal energy is available with decrease in temperature the paramagnetic susceptibility falls rapidly. More complex behaviour may occur if some moments are systematically aligned so as to oppose others, but to a finite magnetic moment: this is *Ferrimagnetism*. The relationship between paramagnetism, ferromagnetism, antiferromagnetism and ferrimagnetism is

illustrated in Figure 2.1.
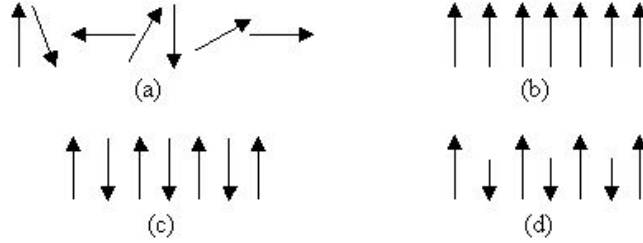
Figure 2.1: Pictorial representation of the relative magnetic spin directions of (a) paramagnetism, (b) ferromagnetism, (c) antiferromagnetism (d) ferrimagnetism. Here the relative magnitude of the spins is shown by the size of the arrow. Taken from Sharpe. [Sha92]

2.6 Modelling magnetic substances

Observing the properties of magnetic substances described in Section 2.5 is difficult as one has to examine the system on the quantum level with all the issues that brings. Instead, it is easier and cheaper to use a statistical model of the substance in a particular configuration[1], such as the Ising model. In this section, we shall concentrate on the energy that the particular configurations have, as this is crucial to the spontaneous magnetisation of the system.

The Ising model consists of a regular array of lattice sites in one, two or three dimensions, in which each site can be occupied in one of two ways. Each lattice site is occupied by an atom that can only exist in one of two spin states: +, with the spin in the direction of the magnetic field B_0, or −, against the field. The potential energy of a single dipole or spin is $-mB_0$ if it is oriented with the field (+), and $+mB_0$ if it is oriented against the field (−), where m is the magnetic moment of an individual atom . Let the state of the jth lattice site be denoted by σ_j, which equals +1 for a + state and −1 for a − state. In terms of these σ_j, the potential energy due to the external field is $-mB_0 \sum \sigma_j$. To simplify this formula, for the later sections we shall use the following definition; $h \equiv mB_0$. It is also assumed that there is an interaction

---

[1]that is, a given temperature, lattice configuration and external magnetic field

ϵ_{ij} between nearest neighbour atoms, which are in sites i and j, where i and j can be $+$ or $-$. In terms of σ_j, ϵ_{ij} can be written as $-J\sigma_i\sigma_j$ where J is the interaction strength between spins. Note that: if the spins are parallel, $\sigma_i\sigma_j$ is positive; if there are anti parallel, $\sigma_i\sigma_j$ is negative. So the nature of the interaction is determined by the sign of J. If $J > 0$, parallel alignments are more stable and the model will describe ferromagnetism. If $J < 0$, an opposed alignment is the more stable and this will lead to antiferromagnetism. These interaction energies are similar to those in chemical bond theory [Sha92].

The total energy of a given configuration of N sites on a particular lattice type and variables, that is a given set of $\{\sigma_i\}$ is then

$$E\left(\sigma_1, \sigma_2, \ldots, \sigma_N\right) = -J \sum_{i,j=1}^{N} \sigma_i\sigma_j - h \sum_{j=1}^{N} \sigma_j \tag{2.8}$$

where the first summation is over all nearest-neighbour pairs. This is the energy expression that characterises the Ising model of a magnetic system. The canonical partition function is the summation over all configurations, weighted by $\exp\left(-E/\left(k_B T\right)\right)$ or

$$Z_N = \sum_{\sigma_1=\pm 1} \cdots \sum_{\sigma_N=\pm 1} \exp\left\{-\frac{E\left(\sigma_1, \sigma_2, \ldots, \sigma_N\right)}{k_B T}\right\} \tag{2.9}$$

where k_B is the Boltzmann constant. The number of terms in this equation is 2^N, because each of the N σ_i sites can take on two values.

In a certain sense, the Ising model is a simpler system than a non-ideal gas or a liquid, because the interacting particles are allowed to be situated only at discrete lattice sites. On the other hand, the model is difficult enough that the exact solution in three dimensions has yet to be found, although the two-dimensional model has been solved exactly in the absence of a magnetic field.

The Ising Model

In this chapter and the next we will look at the development of the solution of the Ising model for various types of lattices. We will first look at the one dimensional model, to show a simpler form of the model being exactly solved. Using this result we will then look at the two-dimensional model on a square lattice. Both the square lattice model and chain model are special in terms of Statistical Mechanics as they are examples of the few models that can be exactly solved. At the end of this chapter we will quickly cover the Ising model in higher dimensions and why these models may never be exactly solved. Also towards the end of the chapter we will quickly discuss other methods of solving and approximating solutions to the Ising model.

3.1   Calculation for one-dimension

To make the development easier we shall start with the one-dimensional model, or chain, with no external magnetic field. This will allow us to see how the interactions in the chain affect the properties. Later we will explain the model in an external field so as to improve the model for its use in applications.
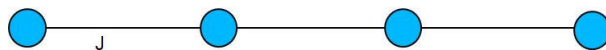
Figure 3.1: Pictorial representation of the one-dimensional Ising chain. Here the circles represent the atoms in the chain, and the lines represent the interparticle interactions with strength J.

In our model, each configuration occurs with the probability proportional to $e^{-\beta E}$, where E is the energy of the system. First, we shall work out the energy, or rather what we shall call the energy which is equal to the enthalpy as there will be no work

done by the system, or each configuration which is given by the following Hamiltonian formula;

$$H = -J \sum_{i=1}^{N-1} \sigma_i \sigma_{i+1} \tag{3.1}$$

where J is a constant. The term being summed is the relative spin configuration of the neighbouring elements.

The canonical partition function (Z_N) is given by

$$Z_N = \sum_{\sigma_1 = \pm 1} \cdots \sum_{\sigma_N = \pm 1} \exp \left\{ \beta J \sum_{i=1}^{N-1} \sigma_i \sigma_{i+1} \right\} \text{ where } \beta = \frac{1}{k_B T}. \tag{3.2}$$

We can see from the equation that the spin of the last position appears only once in the equation and so we have the following formula irrespective of the value of σ_{N-1}:

$$\sum_{\sigma_N = \pm 1} \exp \left\{ \beta J \sigma_{N-1} \sigma_N \right\} = 2 \cosh \beta J. \tag{3.3}$$

Substituting this equation into (2.3) we get

$$Z_N = [2 \cosh \beta J] \, Z_{N-1}$$

Now iterating this operation we can reduce the equation to

$$Z_N = (2 \cosh \beta J)^{N-2} Z_2 \tag{3.4}$$

$$\text{where } Z_2 = \sum_{\sigma_1 = \pm 1} \sum_{\sigma_2 = \pm 1} \exp \left\{ \beta J \sigma_1 \sigma_2 \right\} = 4 \cosh \beta J. \tag{3.5}$$

So combining (3.4) and (3.5) we get the final formula for Z_N:

$$Z_N = 2 \left(2 \cosh \beta J \right)^{N-1} \tag{3.6}$$

The Gibbs free energy of the system is defined by

$$G = -k_B \ln Z_N = -k_B T \left[\ln 2 + (N-1) \ln \left(2 \cosh \beta J \right) \right]$$

In the thermodynamic limit (that is, the limit of the free energy as N tends to infinity),

only the term proportional to N is important and so the formula becomes

$$G = -Nk_BT \ln \left(2 \cosh \beta J \right). \tag{3.7}$$

These calculations give us the exact solution to the model while not in the presence of any magnetic field. We can adapt these formulae and find the solution for a model in an external magnetic field. However, one extra quantity we have to consider is the end effects of this chain when in this field. These end effects could make the equations more difficult than they would need to be, so we shall try to 'fix' the model so that we can ignore the end effects. We can safely 'fix' the end effects as they do not matter in the thermodynamic limit. This is because we assume the chain to be infinitely long in both directions without an end.

One way to 'fix' the problem of the end effects is to consider the chain as a ring, that is, to set periodic boundary conditions. So we shall assume that the Nth spin is connected to the first. Then the Hamiltonian becomes

$$H = -J \sum_{i=1}^{N} \sigma_i \sigma_{i+1} - h \sum_{i=1}^{N} \sigma_i \tag{3.8}$$

where the spin labels run modulo N (that is, $N + i = i$). This formula can then be simplified to

$$H = -\sum_{i=1}^{N} \left[J\sigma_i \sigma_{i+1} + \frac{h}{2} \left(\sigma_i + \sigma_{i+1} \right) \right].$$

The partition function for this new Hamiltonian can be written as

$$
\begin{aligned}
Z_N &= \sum_{\sigma_1 = \pm 1} \cdots \sum_{\sigma_N = \pm 1} \exp \left\{ \beta \sum_{i=1}^{N} \left[J\sigma_i \sigma_{i+1} + \frac{h}{2} \left(\sigma_i + \sigma_{i+1} \right) \right] \right\} \\
\text{or} \quad &= \sum_{\sigma_i} \prod_{i=1}^{N} \exp \left\{ \beta \left[J\sigma_i \sigma_{i+1} + \frac{h}{2} \left(\sigma_i + \sigma_{i+1} \right) \right] \right\}.
\end{aligned} \tag{3.9}
$$

We can see that this formula has the potential of being very large, so at this point we introduce a 2×2 transfer matrix,

$$\mathbf{P} = \begin{bmatrix} P_{11} & P_{1-1} \\ P_{-11} & P_{-1-1} \end{bmatrix} \tag{3.10}$$

where the elements are defined as follows

$$
\begin{aligned}
P_{11} &= e^{\beta(J+h)} \\
P_{-1-1} &= e^{\beta(J-h)} \\
P_{1-1} = P_{-11} &= e^{-\beta J}
\end{aligned}
$$

(3.11)

We shall now use this transfer matrix to describe our partition function in terms of a product of these transfer matrices:

$$
Z_N = \sum_{\langle \sigma_i \rangle} P_{\sigma_1 \sigma_2} P_{\sigma_2 \sigma_3} \cdots P_{\sigma_N \sigma_1} = \operatorname{Tr} \mathbf{P}^N.
$$

(3.12)

We can see that the matrix P can be diagonalised and the eigenvalues λ_1 and λ_2 are the roots of the determinant

$$
\det (\mathbf{P} - \lambda \mathbf{I}) = 0
$$

(3.13)

where $\mathbf{I}$ is the 2×2 identity matrix. Similarly the matrix $\mathbf{P}^N$ has the eigenvalues λ_1^N and λ_2^N, and the trace of $\mathbf{P}^N$ is the sum of the eigenvalues:

$$
Z_N = \lambda_1^N + \lambda_2^N.
$$

(3.14)

The solution of (3.13) is

$$
\lambda_{1,2} = e^{\beta J} \cosh \beta h \pm \sqrt{e^{2\beta J} \sinh^2 \beta h + e^{-2\beta J}}
$$

(3.15)

We note that the eigenvalue associated with the positive root of equation (3.13), λ_1, is always larger in magnitude than the negative root eigenvalue. Now putting this partition function into the equation for the free energy we get

$$
G = -k_B T \ln \left(\lambda_1^N + \lambda_2^N \right) = -k_B T \left\{ N \ln \lambda_1 + \ln \left[1 + \left(\frac{\lambda_2}{\lambda_1} \right)^N \right] \right\}
$$

(3.16)

This approaches $-N k_B T \ln \lambda_1$ as N goes to infinity. Now taking this to the thermodynamic limit,

$$
G = -N k_B T \ln \left[e^{\beta J} \cosh \beta h + \sqrt{e^{2\beta J} \sinh^2 \beta h + e^{-2\beta J}} \right]
$$

In the special case $h = 0$ we obtain the result for the non-magnetic field case, so we have found the general formula. We may compute the magnetisation from

$$m = \langle \sigma_0 \rangle = -\frac{1}{N} \frac{\partial G}{\partial h} = \frac{k_B T}{\lambda_1} \frac{\partial \lambda_1}{\partial h}.$$

After some (straightforward) manipulations we find

$$m = \frac{\sinh \beta h}{\sqrt{\sinh^2 \beta h + e^{-4\beta J}}} \tag{3.17}$$

We see that for $h = 0$ there is no spontaneous magnetisation at any non-zero temperature. However, in the limit of low temperatures

$$\sinh^2 \beta h \gg e^{-4\beta J}$$

for any $h \neq 0$ and only a very small field is needed to produce near saturation of the magnetisation. The zero-field free energy will, in the limit as T approaches zero, approach the value $G = -NJ$ corresponding to completely aligned spins. We could thus say that we have a phase transition at $T = 0$, while for $T \neq 0$ the free energy is an analytic function of its variables.

This is a very interesting result as phase transitions do not occur in the one-dimensional model at a finite temperature. If we look at the model with an initial state we should be able to intuitively illustrate why the phase transition is impossible. The model is set up so $\sigma_i = 1$ if $i \leq l$ and $\sigma_i = -1$ if $i > l$, with the ground state $E_0 = -(N - l)J$. There are $N - l$ such states, all with the same energy $E = E_0 + 2J$. At temperature T the change in free energy will be

$$\Delta G = 2J - k_B T \ln(N - l).$$

This quantity is less than zero for all T greater than zero in the limit where N approaches infinity. The expectation value of the magnetism of the system is zero by using our formula given in (3.17). As the system is at equilibrium if the value of magnetism is zero, the change in free energy would be zero. So we have a contradiction and there can not be a phase transition between the equilibrium phase and the ferromagnetic phase.

3.2   Square lattice two-dimensional calculation

As we have mentioned in Section 1.2 of this chapter, the Norwegian American chemist
Lars Onsager presented the first exact solution of the two-dimensional rectangular
lattice Ising model without an external magnetic field [Ons44]. While this solution
is historically important, it is also quite mathematically formidable. Since Onsager's
solution there have been a number of more transparent solutions presented [Yan52,
Kau49, KW52], and in this section we will look at the method of Schultz, Mattis and
Lieb [SML64, PB94]. This method has been chosen as it follows the same form as the
calculation for the one-dimensional model shown above, and it will relate well to our
later development of the two-dimensional model on the Union Jack lattice.
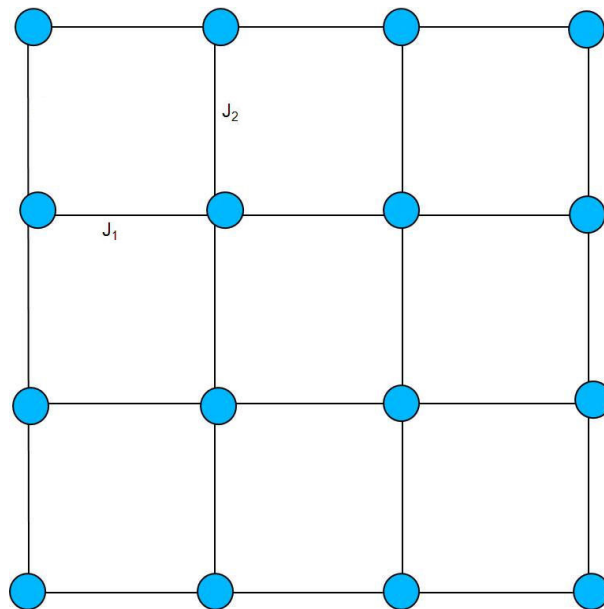
Figure 3.2: Pictorial representation of the two-dimensional Ising model on the square
lattice. Again the lines represent the interparticle interactions, with the horizontal
interactions having strength J_1 and the vertical interactions have strength J_2. The
sites are shown by circles.

3.2.1   Transfer matrix

In the calculation in the one-dimensional model we used a transfer matrix approach
[NM53], and this method can be easily extended to be used for the two-dimensional
model. First let us look at the one-dimensional model and this time formulate the
solution in a slightly different way. First let us rewrite equation (3.9) in the following

way

$$Z = \sum_{\sigma_j = \pm 1} \left(e^{\beta h \sigma_1} e^{K \sigma_1 \sigma_2} \right) \left(e^{\beta h \sigma_2} e^{K \sigma_2 \sigma_3} \right) \dots \left(e^{\beta h \sigma_N} e^{K \sigma_N \sigma_1} \right) \tag{3.18}$$

where $K = \beta J$.

Taking the two orthonormal states $|+1\rangle$ and $|-1\rangle$ we can form a basis and rewrite the Pauli operators as,

$$\sigma_Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \sigma^+ = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad \sigma^- = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \tag{3.19}$$

with $\sigma_X = \sigma^+ + \sigma^-$ and $\sigma_Y = -i(\sigma^+ - \sigma^-)$. The Boltzmann weight $\exp\{\beta h \sigma_j\}$ can be expressed as a diagonal matrix, $\mathbf{V}_1$, in this basis:

$$\langle +1 | \mathbf{V}_1 | + 1 \rangle = e^{\beta h}, \ \langle -1 | \mathbf{V}_1 | - 1 \rangle = e^{-\beta h}$$

or

$$\mathbf{V}_1 = \exp\left(\beta h \sigma_Z \right). \tag{3.20}$$

We can also define the operator $\mathbf{V}_2$ corresponding to the nearest-neighbour coupling by its matrix elements in this basis:

$$\langle +1 | \mathbf{V}_2 | + 1 \rangle = \langle -1 | \mathbf{V}_2 | - 1 \rangle = e^{K}$$
$$\langle +1 | \mathbf{V}_2 | - 1 \rangle = \langle -1 | \mathbf{V}_2 | + 1 \rangle = e^{-K}.$$

Therefore,

$$\mathbf{V}_2 = e^{K} \mathbf{1} + e^{-K} \sigma_X = A(K) \exp\{K^* \sigma_X\} \tag{3.21}$$

where in the second step we have used the fact that $(\sigma_X)^{2n} = \mathbf{1}$. The constants $A(K)$ and K^* are determined from the equations

$$A \cosh K^* = e^{K}$$
$$A \sinh K^* = e^{-K} \tag{3.22}$$

or $\tanh K^* = \exp(-2K)$, $A = \sqrt{2 \sinh\{2K\}}$. From these results we can rewrite the

partition function as:

$$
\begin{aligned}
Z &= \sum_{\{\mu=+1,-1\}} \langle \mu_1 | \mathbf{V}_1 | \mu_2 \rangle \langle \mu_2 | \mathbf{V}_2 | \mu_3 \rangle \langle \mu_3 | \mathbf{V}_1 | \mu_4 \rangle \dots \langle \mu_{2N} | \mathbf{V}_2 | \mu_1 \rangle \\
&= \mathrm{Tr}\,(\mathbf{V}_1 \mathbf{V}_2)^N = \mathrm{Tr}\,(\mathbf{V}_2^{1/2} \mathbf{V}_1 \mathbf{V}_2^{1/2})^N = \lambda_1^N + \lambda_2^N
\end{aligned}
\tag{3.23}
$$

where λ_1 and λ_2 are the two eigenvalues of the Hermitian operator

$$
\mathbf{V} = (\mathbf{V}_2^{1/2} \mathbf{V}_1 \mathbf{V}_2^{1/2}) = \sqrt{2\sinh(2K)}\, e^{K^* \sigma_X/2} e^{\beta h \sigma_Z} e^{K^* \sigma_X/2}.
\tag{3.24}
$$

We have managed to get to this symmetric form of the transfer matrix $\mathbf{V}$ by using the property of the invariance of the trace of a product of matrices under a cyclic permutation of factors.

It is interesting to note that in this method we have taken a one-dimensional classical statistics problem and transformed it into a zero-dimensional quantum-mechanical ground state problem. This is not the only time this transformation can be achieved and the result is quite general [Suz76, Suz85]. There is a correspondence between the ground state of quantum Hamiltonians in $d-1$ dimensions and classical partition functions in d dimensions. This can sometimes be exploited, for example, in numerical simulations of quantum-statistical models.

We can now generalise this method to the two-dimensional Ising model. This time consider an $M \times M$ square lattice with periodic boundary conditions and the Hamiltonian

$$
H = -J \sum_{r,c} \sigma_{r,c}\sigma_{r+1,c} - J \sum_{r,c} \sigma_{r,c}\sigma_{r,c+1}
\tag{3.25}
$$

where the label r refers to rows, c to columns, and $\sigma_{r+M,c} = \sigma_{r,c+M} = \sigma_{r,c}$. Here the first sum contains only interactions in column c and the second sum is the coupling between neighbouring columns and will lead to a non-diagonal factor in the complete transfer matrix.

Now as we did with the one-dimensional case, we introduce the 2^M basis states

$$
|\mu\rangle \equiv |\mu_1\rangle |\mu_2\rangle \dots |\mu_M\rangle
\tag{3.26}
$$

with $\mu_j = \pm 1$ and M sets of Pauli operators $(\sigma_{jX}, \sigma_{jY}, \sigma_{jZ})$ which act on the jth state

in the product, that is,

$$\sigma_{jZ} |\mu_1, \mu_2, \ldots \mu_j, \ldots \mu_M\rangle = \mu_j |\mu_1, \mu_2, \ldots \mu_j \ldots \mu_M\rangle$$

$$\sigma_j^+ |\mu_1, \mu_2, \ldots \mu_j, \ldots \mu_M\rangle = \delta_{\mu_j, -1} |\mu_1, \mu_2, \ldots \mu_j + 2 \ldots \mu_M\rangle$$

$$\sigma_j^- |\mu_1, \mu_2, \ldots \mu_j, \ldots \mu_M\rangle = \delta_{\mu_j, 1} |\mu_1, \mu_2, \ldots \mu_j - 2 \ldots \mu_M\rangle \tag{3.27}$$

Moreover, we impose the commutation relations $[\sigma_{j\alpha}, \sigma_{m\beta}] = 0$, $\alpha, \beta = X, Y, Z$ for $j \neq m$. For $j = m$ the usual Pauli matrix commutation relations apply.

We can think of the index μ_j as the orientation of the jth spin in a given column. From this we can see that the Boltzmann factors $\exp\{K \sum_r \sigma_{r,c}\sigma_{r+1,c}\}$ are given by the matrix elements of the operator $\mathbf{V}_1 = \exp\{K \sum_j \sigma_{jZ}\sigma_{(j+1)Z}\}$. Similarly, the matrix element

$$\langle \{\mu\} | \mathbf{V}_2 | \{\mu'\} \rangle = \left\langle \mu_M, \mu_{M-1}, \ldots, \mu_1 \left| \prod_{j=1}^{M} \left(e^K \mathbf{1} + e^{-K}\sigma_{jX}\right) \right| \mu_1', \mu_2', \ldots, \mu_M' \right\rangle$$

$$= \exp\{(M - 2n)K\} \tag{3.28}$$

where n of the indices $\{\mu'\}$ differ from the corresponding entries in $\{\mu\}$. Then in a zero magnetic field the partition function of the two-dimensional model is given by

$$Z = \sum_{\{\mu_1\}, \{\mu_2\}, \ldots, \{\mu_M\}} \langle \mu_1 | \mathbf{V}_1 | \mu_2 \rangle \langle \mu_2 | \mathbf{V}_2 | \mu_3 \rangle \langle \mu_3 | \mathbf{V}_1 | \mu_4 \rangle \ldots \langle \mu_M | \mathbf{V}_2 | \mu_1 \rangle$$

$$= \mathrm{Tr}\,(\mathbf{V}_1 \mathbf{V}_2)^M = \mathrm{Tr}\left(\mathbf{V}_2^{1/2}\mathbf{V}_1\mathbf{V}_2^{1/2}\right)^M. \tag{3.29}$$

Above the sum over each $\{\mu_J\}$ is over the entire set of 2^M basis states. We can then write, using 3.21 and 3.22

$$\mathbf{V}_2 = (2\sinh 2K)^{M/2}\exp\left\{K^*\sum_{j=1}^{M}\sigma_{jX}\right\} \tag{3.30}$$

and as such the calculation of the partition function has been reduced to finding the

largest eigenvalue of the Hermitian operator

$$
\begin{aligned}
\mathbf{V} &= \mathbf{V}_2^{1/2}\mathbf{V}_1\mathbf{V}_2^{1/2} \\
&= (2\sinh 2K)^{M/2}\exp\left\{K^*\sum_{j=1}^M \sigma_{jX}\right\}\exp\left\{K^*\sum_{j=1}^M \sigma_{jZ}\sigma_{j+1,Z}\right\}\exp\left\{K^*\sum_{j=1}^M \sigma_{jX}\right\}.
\end{aligned}
$$

(3.31)

This is still a non-trivial task since the factors do not commute with each other, and because the matrix $\mathbf{V}$ becomes infinite-dimensional in the thermodynamic limit.

3.2.2 Transformation to an interacting fermion problem

We now perform a rotation of the spin operators for future simplicity and let $\sigma_{jZ} \to -\sigma_{jX}$, $\sigma_{jX} \to \sigma_{jZ}$ for all $j = 1, \ldots, M$. The eigenvalue are invariant under this rotation. Using $\sigma_{jZ} = 2\sigma_j^+\sigma_j^- - \mathbf{1}$ and $\sigma_{jX} = \sigma_j^+ + \sigma_j^-$ we arrive at

$$
\begin{aligned}
\mathbf{V}_1 &= \exp\left\{K\sum_{j=1}^K \left(\sigma_j^+ + \sigma_j^-\right)\left(\sigma_{j+1}^+ + \sigma_{j+1}^-\right)\right\} \\
\mathbf{V}_2 &= (2\sinh 2K)^{M/2}\exp\left\{2K^*\sum_{j=1}^M \left(\sigma_j^+\sigma_j^- - \frac{1}{2}\mathbf{1}\right)\right\}.
\end{aligned}
$$

(3.32)

Now following the method of Schultz *et al* [SML64] we perform a Jordan-Wigner transformation, which converts the Pauli operators into fermion operators (see [PB94] for a discussion of second quantisation). This step is necessary as some of the subsequent canonical transformation are not possible for angular momentum operators. We write

$$
\begin{aligned}
\sigma_j^+ &= \exp\left\{\pi i\sum_{m=1}^{j-1} c_m^\dagger c_m\right\}c_j^\dagger \\
\sigma_j^- &= c_j\exp\left\{-\pi i\sum_{m=1}^{j-1} c_m^\dagger c_m\right\} = \exp\left\{\pi i\sum_{m=1}^{j-1} c_m^\dagger c_m\right\}c
\end{aligned}
$$

(3.33)

where the operators c, $c^\dagger$ obey the commutation relations

$$\left| c_j, c_m^\dagger \right|_+ = c_j c_m^\dagger + c_m^\dagger c_j = \delta_{jm}$$

$$\left| c_j, c_m \right|_+ = \left| c_j^\dagger, c_m^\dagger \right|_+ = 0.$$

The operator $c_m^\dagger c_m$ is the fermion number operator for site m with integer eigenvalues 0 and 1.

We can now express the operators $\mathbf{V}_1$ and $\mathbf{V}_2$ in terms of fermion operators using equation (3.33). The operator $\mathbf{V}_2$ presents no difficulties and is given by

$$\mathbf{V}_2 = (2\sinh 2K)^{M/2} \exp\left\{ 2K^* \sum_{j=1}^{M} \left(c_j^\dagger c_j - \frac{1}{2} \right) \right\}. \tag{3.34}$$

In the case of $\mathbf{V}_1$, there is a slight difficulty due to the periodic boundary condition. For $j \neq M$ we note that the term

$$\left(\sigma_j^+ + \sigma_j^- \right) \left(\sigma_{j+1}^+ + \sigma_{j+1}^- \right) = c_j^\dagger c_{j+1}^\dagger + c_j^\dagger c_{j+1} + c_{j+1}^\dagger c_j + c_{j+1} c_j.$$

For the specific case $j = M$,

$$
\begin{aligned}
\left(\sigma_M^+ + \sigma_M^- \right) \left(\sigma_1^+ + \sigma_1^- \right) &= \exp\left\{ \pi i \sum_{j=1}^{M-1} c_j^\dagger c_j \right\} c_M^\dagger \left(c_1^\dagger + c_1 \right) + \exp\left\{ \pi i \sum_{j=1}^{M-1} c_j^\dagger c_j \right\} c_M \left(c_1^\dagger + c_1 \right) \\
&= \exp\left\{ \pi i \sum_{j=1}^{M} c_j^\dagger c_j \right\} \left[\exp \pi i c_M^\dagger c_M \left(c_M^\dagger + c_M \right) \left(c_1^\dagger + c_1 \right) \right] \\
&= (-1)^n \left(c_M - c_M^\dagger \right) \left(c_1^\dagger + c_1 \right)
\end{aligned}
$$

where $n = \sum_j c_j^\dagger c_j$ is the total fermion number operator. The operator n commutes with $\mathbf{V}_2$ but not with $\mathbf{V}_1$. On the other hand, $(-1)^n$ commutes with both $\mathbf{V}_1$, and $\mathbf{V}_2$ as the various terms in $\mathbf{V}_1$ change the total fermion number by 0 or ± 2. So we can write $\mathbf{V}_1$ in a universal way by considering the subspaces of even and odd total number of fermions, that is,

$$\mathbf{V}_1 = \exp\left\{ K \sum_{j=1}^{M} \left(c_j^\dagger - c_j \right) \left(c_{j+1}^\dagger + c_{j+1} \right) \right\} \tag{3.35}$$

where

$$
\begin{array}{lll}
c_{M+1} \equiv -c_1, & c_{M+1}^\dagger \equiv -c_1^\dagger & \text{for } n \text{ even} \\
c_{M+1} \equiv c_1, & c_{M+1}^\dagger \equiv c_1^\dagger & \text{for } n \text{ odd.}
\end{array}
\tag{3.36}
$$

This choice of boundary condition on the fermion creation and annihilation operators allows us to recover translational invariance, so we can carry out the canonical transformation

$$
\begin{aligned}
a_q &= \frac{1}{\sqrt{M}} \sum_{j=1}^{M} c_j e^{-iqj} \\
a_q^\dagger &= \frac{1}{\sqrt{M}} \sum_{j=1}^{M} c_j^\dagger e^{iqj}
\end{aligned}
\tag{3.37}
$$

with inverse

$$
\begin{aligned}
c_j &= \frac{1}{\sqrt{M}} \sum_{q} a_q e^{iqj} \\
c_j^\dagger &= \frac{1}{\sqrt{M}} \sum_{q} a_q^\dagger e^{-iqj}.
\end{aligned}
\tag{3.38}
$$

where we take $q = j\pi/M$ to reproduce the boundary conditions in (3.36). Substituting this into the equations for $\mathbf{V}_2$ and $\mathbf{V}_1$ we get for n even,

$$
\begin{aligned}
\mathbf{V}_2 &= (2 \sinh 2K)^{M/2} \exp\left\{ 2K^* \sum_{q>0} \left(a_q^\dagger a_q + a_{-q}^\dagger a_{-q} - 1 \right) \right\} \\
&= (2 \sinh 2K)^{M/2} \prod_{q>0} \mathbf{V}_{2q}
\end{aligned}
\tag{3.39}
$$

and

$$
\begin{aligned}
\mathbf{V}_1 &= \exp\left\{ 2K \sum_{q>0} \left[\cos q \left(a_q^\dagger a_q + a_{-q}^\dagger a_{-q} \right) - i \sin q \left(a_q^\dagger a_{-q}^\dagger + a_q a_{-q} \right) \right] \right\} \\
&= \prod_{q>0} \mathbf{V}_1 q.
\end{aligned}
\tag{3.40}
$$

In (3.39) and (3.40), the terms corresponding to q and $-q$ have been combined. The resultant operators can be written as products and we can then see that bilinear operators with different wave vectors commute. This in turn allows great simplification

and the eigenvalues of the transfer matrix can be written as a product of eigenvalues of at most 4×4 matrices. For odd n we need the operators $\mathbf{V}_{1q}$ and $\mathbf{V}_{2q}$ for $q = \pi$ and $q = 0$. These are given by

$$
\begin{aligned}
\mathbf{V}_{10} &= \exp\left\{2K a_0^\dagger a_0\right\} & \mathbf{V}_{20} &= \exp\left\{2K^*\left(a_0^\dagger a_0 - \tfrac{1}{2}\right)\right\} \\
\mathbf{V}_{1\pi} &= \exp\left\{-2K a_\pi^\dagger a_\pi\right\} & \mathbf{V}_{2\pi} &= \exp\left\{2K^*\left(a_\pi^\dagger a_\pi - \tfrac{1}{2}\right)\right\}
\end{aligned}
\tag{3.41}
$$

which are already in diagonal form and commute with each other.

3.2.3   Calculation of eigenvalues

We now go on to calculate the eigenvalues of the operator

$$
\mathbf{V}_q = \mathbf{V}_{2q}^{1/2} \mathbf{V}_{1q} \mathbf{V}_{2q}^{1/2}
$$

for $q \neq 0$ and $q \neq \pi$. As we are dealing with fermions, we only have four possible states: $|0\rangle$, $a_q^\dagger|0\rangle$, $a_{-q}^\dagger|0\rangle$ and $a_q^\dagger a_{-q}^\dagger|0\rangle$, where $|0\rangle$ is the zero particle state defined by $a_q|0\rangle = a_{-q}|0\rangle = 0$. These states are eigenstates of $\mathbf{V}_2$. Since the operator $\mathbf{V}_1$ has non-zero off-diagonal matrix elements only when two states differ by two fermions, the problem reduces to finding the eigenvalues of $\mathbf{V}_q$ in the basis $|0\rangle$ and $|2\rangle = a_q^\dagger a_{-q}^\dagger|0\rangle$. We note that

$$
\mathbf{V}_{1q} a_{\pm q}^\dagger|0\rangle = \exp\left\{2K\cos q\right\} a_{\pm q}^\dagger|0\rangle
\tag{3.42}
$$

and

$$
\begin{aligned}
\mathbf{V}_{2q}^{1/2}|0\rangle &= \exp\left\{-K^*\right\}|0\rangle \\
\mathbf{V}_{2q}^{1/2}|2\rangle &= \exp\left\{K^*\right\}|2\rangle.
\end{aligned}
\tag{3.43}
$$

To obtain the matrix elements of $\mathbf{V}_{1q}$ in the basis $|0\rangle$, $|2\rangle$, we let

$$
\mathbf{V}_{1q}|0\rangle = \alpha(K)|0\rangle + \beta(K)|2\rangle.
$$

When we differentiate this with respect to K, we get

$$
\begin{aligned}
\frac{d\alpha}{dK} &= 2i\beta(K)\sin q \\
\frac{d\beta}{dK} &= 4\beta(K)\cos q - 2i\alpha(K)\sin q.
\end{aligned}
\tag{3.44}
$$

Subject to the boundary conditions $\alpha(0) = 1$, $\beta(0) = 0$, we solve these equations to find

$$
\begin{aligned}
\langle 0 \left| \mathbf{V}_{1q} \right| 0 \rangle &= \alpha(K) = e^{2K\cos q}\left(\cosh 2K - \sinh 2K \cos q\right) \\
\langle 2 \left| \mathbf{V}_{1q} \right| 0 \rangle &= \beta(K) = -ie^{2K\cos q}\sinh 2K \sin q.
\end{aligned}
\tag{3.45}
$$

Using this method the matrix elements for $\langle 2 \left| \mathbf{V}_{1q} \right| 2 \rangle$ and $\langle 0 \left| \mathbf{V}_{1q} \right| 2 \rangle = \langle 2 \left| \mathbf{V}_{1q} \right| 0 \rangle^*$ and obtain the matrix

$$
\mathbf{V}_{1q} = e^{2K\cos q}\left[
\begin{array}{cc}
\cosh 2K - \sinh 2K \cos q & i\sinh 2K \sin q \\
-i\sinh 2K \sin q & \cosh 2K + \sinh 2K \cos q
\end{array}
\right]
\tag{3.46}
$$

and

$$
\mathbf{V}_q = \left[
\begin{array}{cc}
\exp\{-K^*\} & 0 \\
0 & \exp\{K^*\}
\end{array}
\right]
[\mathbf{V}_{1q}]
\left[
\begin{array}{cc}
\exp\{-K^*\} & 0 \\
0 & \exp\{K^*\}
\end{array}
\right].
\tag{3.47}
$$

The eigenvalues of this matrix are easily determined and can be given in the form

$$
\lambda_q^{\pm} = \exp\{2K\cos q \pm \epsilon(q)\}
\tag{3.48}
$$

where after a bit of algebra $\epsilon(q)$ can be defined by

$$
\cosh \epsilon(q) = \cosh 2K \cosh 2K^* + \cos q \sinh 2K \sinh 2K^*.
$$

By convention we choose $\epsilon(q) \geq 0$. It can be seen that the minimum of the right hand side of this equation occurs as $q \to \pi$ and that for all q

$$
\epsilon(q) > \epsilon_{\min} = \lim_{q \to \pi} \epsilon(q) = 2\left|K - K^*\right|
\tag{3.49}
$$

and also

$$\lim_{q \to 0} \epsilon(q) = 2 \left(K + K^* \right). \tag{3.50}$$

Now we combine this information and consider the subspace on an even number of fermions. The wave vectors in this case do not include $q = 0$ or $q = \pi$, and it can be seen that the largest eigenvalue of $\mathbf{V}_q$ for each q is λ_q^+. Thus the largest eigenvalue in this subspace is given by

$$
\begin{aligned}
\Lambda_e &= (2 \sinh 2K)^{M/2} \prod_{q>0} \lambda_q^+ \\
&= (2 \sinh 2K)^{M/2} \exp\left\{ \sum_{q>0} [2\cos q + \epsilon(q)] \right\} \\
&= (2 \sinh 2K)^{M/2} \exp\left\{ \frac{1}{2} \sum_q \epsilon(q) \right\}
\end{aligned}
\tag{3.51}
$$

where we have used that $\sum_q \cos q = 0$ and have extended the summation over the entire range, $-\pi < q < \pi$.

It is slightly more difficult to work with the odd subspace. As before for $q \neq 0$ and $q \neq \pi$ the maximum possible eigenvalue is λ_q^+. The corresponding eigenstates are all states with $(-1)^n = -1$. So to make the overall state have the property $(-1)^n = -1$, we occupy the $q = 0$ state and leave the $q = \pi$ state empty, and as such obtain a contribution of $(2 \sinh 2K)^{M/2} \exp\{2K\}$ to the eigenvalue Λ_o. So the largest eigenvalue in the odd subspace is

$$\Lambda_o = (2 \sinh 2K)^{M/2} \exp\left\{ 2K + \frac{1}{2} \sum_{q \neq 0, \pi} \epsilon(q) \right\}. \tag{3.52}$$

With further consideration of these results, it can be shown that Λ_e and Λ_o are degenerate. The critical temperature of the two-dimensional Ising model is given by the equation $K = K^*$ or by expression

$$\sinh \frac{2J}{k_B T_c} = 1 \tag{3.53}$$

which is approximately $k_B T_c / J = 2.269185\ldots$

The degeneracy of the two largest eigenvalues is negligible in the dimensionless free

energy and only contributes an additive term of $\ln 2$. So for any temperature the free energy is given by

$$
\begin{aligned}
\frac{\beta G(0,T)}{M^2} &= \beta g(0,T) = -\frac{1}{2}\ln\left(2\sinh 2K\right) - \frac{1}{2M}\sum_q \epsilon(q) \\
&= -\frac{1}{2}\ln\left(2\sinh 2K\right) - \frac{1}{4\pi}\int_{-\pi}^{\pi} dq\, \epsilon(q)
\end{aligned}
\tag{3.54}
$$

where the sum over the wave vectors has been converted to an integral.

3.2.4 Spontaneous magnetisation

While other thermodynamic functions can be derived, now we have the free energy of the system, which can be simplified with a bit more algebra. A full derivation of this result along with the derivation of the specific heat capacity of the model can be found in [PB94]. It is also possible to extend the theory discussed in this section to calculate the spontaneous magnetisation, as in [SML64]. The result is

$$
\begin{aligned}
m_0(T) &= -\lim_{h\to 0}\frac{\partial}{\partial h}g(h,T) \\
&= \begin{cases} \left[1 - \frac{\left(1-\tanh^2\beta J\right)^4}{16\tanh^4\beta J}\right]^{\frac{1}{8}} & T < T_c \\ 0 & T > T_c \end{cases}
\end{aligned}
\tag{3.55}
$$

As $T \to T_c$, the limiting form of the spontaneous magnetisation is given by

$$
m_0(T) \approx (T_c - T)^{1/8} \equiv (T_c - T)^{\beta}.
\tag{3.56}
$$

At the critical point, the order parameter has a power law singularity.

3.3 Triangular lattice two-dimensional model

Now we add diagonals in one direction to the square lattice to obtain the triangular lattice seen in Figure 3.3. This lattice configuration will be used as our "test" lattice in the later chapters. In this section we will state the results for the two site interaction spontaneous magnetisation and then go on to look at the three site interaction result.
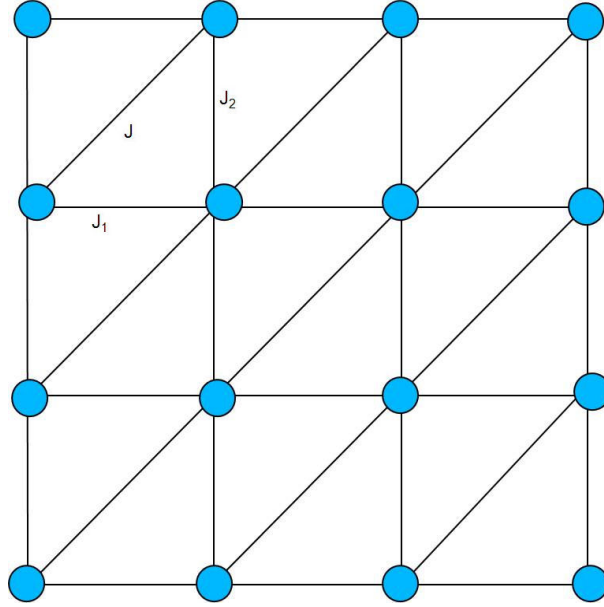
Figure 3.3: Pictorial representation of the two-dimensional Ising model on the Triangular lattice. Interparticle interactions are shown here with lines, and have interactions strength J_1 horizontally, J_2 vertically and J diagonally. Sites are shown with circles.

3.3.1   Two-site interactions on a triangular lattice

Stephenson in his paper [Ste64] shows the development of the two site interaction result. As this development is similar to the development for the square lattice result, here we will state the final result for the spontaneous magnetisation in (3.57):

$$M = \begin{cases} \left(1 - k_1^2\right)^{\frac{1}{8}} & T < T_c \\ 0 & T \geq T_c \end{cases} , \tag{3.57}$$

where

$$k_1^2 = \frac{\left[\left(1 - v_1^2\right)\left(1 - v_2^2\right)\left(1 - v_3^2\right)\right]^2}{16\left(1 + v_1 v_2 v_3\right)\left(v_1 + v_2 v_3\right)\left(v_2 + v_3 v_1\right)\left(v_3 + v_1 v_2\right)},$$

$$v_1 = \tanh \beta J_1, \quad v_2 = \tanh \beta J_2, \quad v_3 = \tanh \beta J. \tag{3.58}$$

Similar derivations can be seen in [Pot52, Gre62].

3.3.2   Three-site triplet interactions on a triangular lattice

Having a result for the two-site nearest neighbour interactions, we now move on to look at the three site triplet interactions. When we consider these interactions the

Hamiltonian (3.25) becomes

$$H = -J' \sum \sigma_x \sigma_y \sigma_z - J \sum \sigma_x \sigma_y - B \sum \sigma_x, \qquad (3.59)$$

where the first summation is over all $2N$ three-site triplet interactions, the second is over all $3N$ two-site nearest neighbour interactions and the third is over all N sites of the lattice. When any of the two of B, J, J' are zero, the free energy of the system can be evaluated exactly. In the case where $B = J = 0$ we obtain the pure three-spin model, whose free energy was obtained by Baxter and Wu in [BW73].

In his approach to calculate the spontaneous magnetisation of the three-site interactions, Baxter [Bax75] considers the case when $H = J' = 0$. This allows a return to the normal two-site interaction triangular Ising model. For this he introduces a new parameter for the three-site spontaneous magnetisation, M_3, and defines $\mathcal{R}_\infty$ to be the ratio between the three-site correlator and the magnetisation in the infinite lattice case, i.e.

$$\mathcal{R}_\infty = \frac{M_3}{M}.$$

Here the three-site correlator is the interaction between a triplet of sites σ_x, σ_y σ_z on a triangular face, $\langle \sigma_x \, \sigma_y \, \sigma_z \rangle$. From (62) of [Bax75] we have that

$$
\begin{aligned}
\mathcal{R}_\infty &= \frac{1}{2} \left(v_1 + v_1^{-1} + v_2 + v_2^{-1} + v_3 + v_3^{-1} \right) \\
&\quad - \frac{1}{2} \left(v_1 v_2 v_3 \right)^{-1} \left[(1 + v_1 v_2 v_3) (v_1 + v_2 v_3) (v_2 + v_3 v_1) (v_3 + v_1 v_2) \right]^{\frac{1}{2}}. \quad (3.60)
\end{aligned}
$$

This then means we can evaluate the spontaneous magnetisation of the three-site interactions as

$$M_3 = \begin{cases} M\mathcal{R} & T < T_c, \\ 0 & T \geq T_c. \end{cases} \qquad (3.61)$$

3.3.3 Three-site triplet interactions on a square lattice

Now consider the case when $J = v_3 = 0$. By removing the diagonal interactions we have the square lattice. M_3 now becomes the three-spin magnetisation around a corner of the square lattice. Using the equations above we can obtain

$$M_3 = M \left[1 - \frac{4e^{-4\beta J_1 - 4\beta J_2}}{\left(1 - e^{-4\beta J_1} \right) \left(1 - e^{-4\beta J_2} \right)} \right], \qquad (3.62)$$

where M is the spontaneous magnetisation of the square lattice. Other three-spin magnetisations have been evaluated by Pink [Pin68].

The Union Jack Lattice

Having discussed the two-dimensional model both on the square and triangular lattices in the last chapter, in this chapter we will focus on the Union Jack lattice or centred squared lattice. We obtain the Union Jack lattice by adding alternate diagonals to the squares of the square lattice, as shown in Figure 4.1. Here the the nearest neighbour
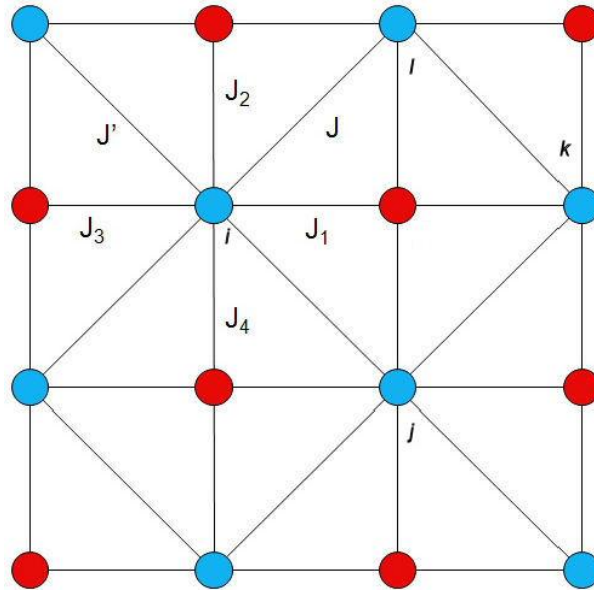
Figure 4.1: Pictorial representation of the two dimensional Ising model on the Union Jack lattice. Here, as before, the interparticle interactions are shown with lines. The red circles represent τ-sites which interact with four particles. The blue circles represent σ-sites with eight interparicle interactions. The various interaction strengths are shown on the different interactions.

interaction strengths can be one of six values J_1, J_2, J_3, J_4, J, J'. The Hamiltonian

for this lattice is

$$
\begin{aligned}
H &= \sum \left[\sigma_{i,j} \left(-J_1 \sigma_{i+1,j} - J_2 \sigma_{i,j+1} - J_3 \sigma_{i-1,j} - J_4 \sigma_{i,j-1} \right) \right] \\
&+ \sum \left[\sigma_{2i-1,2j-1} \left(-J \left(\sigma_{i+1,j+1} + \sigma_{i-1,j-1} \right) - J' \left(\sigma_{i-1,j+1} + \sigma_{i+1,j-1} \right) \right) \right] \\
&- B \sum \sigma_{i,j}.
\end{aligned}
\tag{4.1}
$$

It is clear to see that this lattice $(\mathcal{L}_l)$ can be made of two sublattices, indicated by red and blue circles. The sublattice $\mathcal{L}''$ contains the τ sites, red circles, with four nearest neighbour interactions and the sublattice $\mathcal{L}'$ contains the σ sites, blue circles, with eight nearest neighbour interactions. The partition function on $\mathcal{L}_l$ is

$$
Z_l = \sum_{\sigma_1 = \pm 1} \cdots \sum_{\sigma_N = \pm 1} \exp \left(\beta \sum_{i,j=1}^{N-1} J_r \sigma_i \sigma_j \right).
\tag{4.2}
$$

Vaks, Larkin and Ovchinnikov [VLO66] first considered the Union Jack lattice Ising model as a system exhibiting a re-entrant transition in the presence of competing interactions. They considered a system where the interactions were symmetric, and obtained its free energy and a sublattice two-spin correlation function. This was later generalised by Sacco and Wu [SW75] as a 32-vertex model on a triangular lattice.

Wu and Lin presented a simple formulation for the general Union Jack lattice model [WL87, WL89] and showed that it is equivalent to a free fermion model [FW70]. Using this equivalence they obtained the free energy and the eight site interaction sublattice spontaneous magnetisation. This was extended for symmetric interactions by Lin and Wang [LW87] to produce a result for four interaction sublattice. Wu and Lin in their 1989 paper [WL89] then extend the result further to produce results for both sublattices in the general interaction model.

4.1 The Valks *et al* result

Although we will be presenting the general Union Jack lattice result from Wu and Lin [WL87, WL89] in this chapter, it is useful to look at the results from the paper by Valks *et al* [VLO66]. In their paper, they use a symmetric interaction model where the J_r, $r = 1, 2, 3, 4$ are equal and $J = J'$. They showed that the phase of the system

on the σ-sublattice can be determined by two parameters α_1 and α_2, given by (4.3).

$$\alpha_1 = \frac{e^{2K}\left(\cosh 4K_1 - e^{-2K}\right)}{\left(1 + e^{-2K}\right)},$$

$$\alpha_2 = \frac{e^{-2K}\left(1 - e^{-2K}\right)}{\left(\cosh 4K_1 + e^{-2K}\right)} \tag{4.3}$$

where $K = J/k_B T$ and $K_1 = J_1/k_B T$. The critical temperature T_c is determined when $\alpha_1 = 1$, which has solutions when $-|J_1| < J$. A second critical temperature T_c^* is determined by $\alpha_2 = -1$, which has solutions when $J_2 < -0.907\,|J_1|$. This is shown in Figure 4.2. With further analysis of the formulas given by Valks *et al* one
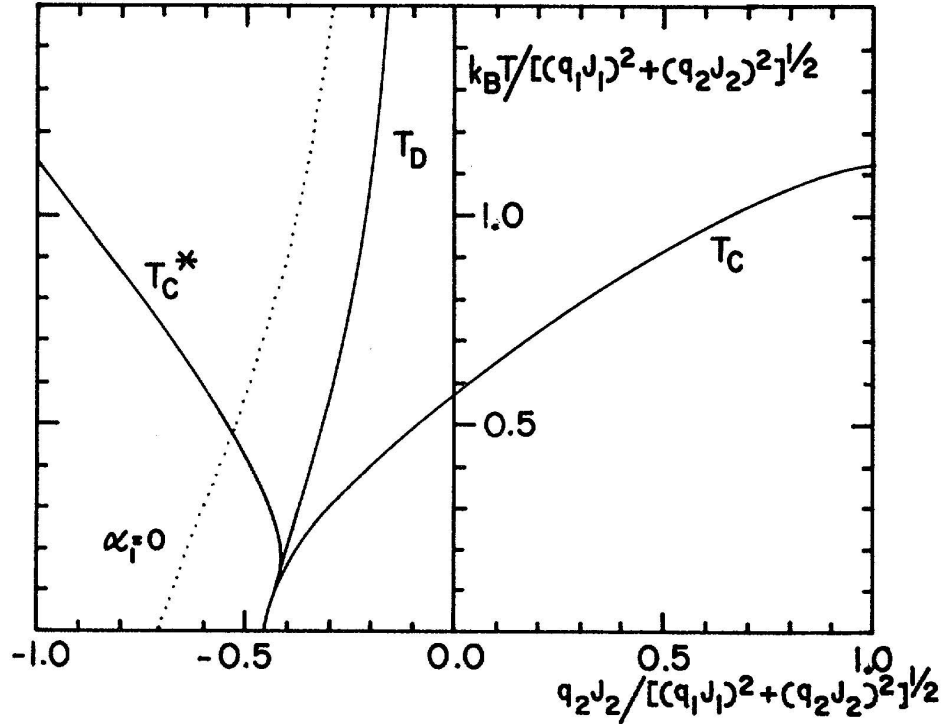
Figure 4.2: Phase diagram of the Vaks *et al.* result on the Union Jack lattice Ising model showing the graphs of the critical temperature T_c, disorder point T_D, and second critical temperature T_c^*. Below T_c the system is in a ferromagnetic phase. Below T_c^* the system is in an ordered antiferromagnetic phase. Above these critical temperatures the system is in a disorder phase. Taken from [Ste70].

can classify the phases of the system. When $-|J_1| < J_2 < 0$, there is an ordered ferromagnetic phase below T_c. Above T_c, there is disordered ferromagnetic phase up to a disorder temperature T_D, determined by $\alpha_1 = -\alpha_2$. At temperatures above T_D, there is a disordered antiferromagnetic phase. This region extends to $T = \infty$, if J satisfies

$-0.907\,|J_1| < J < 0$. However, if $-\,|J_1| < J < -0.907\,|J_1|$, there is an intervening ordered antiferromagnetic phase between the lower and upper critical temperatures T_c^*.

4.2   Computation on the Union Jack lattice

Now we move on to look at the exact result for the general anisotropic Ising model on the Union Jack lattice from Wu and Lin [WL87, WL89]. As the development is similar to that of the two-dimensional square lattice model shown in section 3.2, we will present the result only.

By summing over all the $N/2$ τ sites on $\mathcal{L}''$ we can rewrite the partition function Z_l as limit variables

$$Z_l = \sum_{\tau_1=\pm1} \cdots \sum_{\tau_{\frac{N}{2}}=\pm1} \left[\prod_{i,j,k,l} \omega\,(\sigma_i, \sigma_j, \sigma_k, \sigma_l) \right] \tag{4.4}$$

where the product is over all $N/2$ faces of the sublattice $\mathcal{L}'$, each surrounded by four spins i, j, k, l. Here the Boltzmann factor $\omega(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ is obtained by dividing each of the diagonal interactions $-J$ and $-J'$ into halves, one for each of the two adjacent faces. This leads to

$$\omega\,(a,b,c,d) = 2 \exp\left[\frac{\beta J\,(ab+cd)}{2} + \frac{\beta J'\,(ad+bc)}{2} \right] \cosh\,(a\beta J_1 + b\beta J_2 + c\beta J_3 + d\beta J_4)\,. \tag{4.5}$$

This equation with its four spin variables gives us sixteen separate states which by

symmetry can be reduced to eight distinct expressions:

$$
\begin{aligned}
\omega_1 &= \omega\left(+\,+\,+\,+\right) = 2e^{\beta J + \beta J'}\cosh\left(\beta\left(J_1 + J_2 + J_3 + J_4\right)\right) \\
\omega_2 &= \omega\left(+\,-\,+\,-\right) = 2e^{-\beta J - \beta J'}\cosh\left(\beta\left(J_1 - J_2 + J_3 - J_4\right)\right) \\
\omega_3 &= \omega\left(+\,-\,-\,+\right) = 2e^{-\beta J + \beta J'}\cosh\left(\beta\left(J_1 - J_2 - J_3 + J_4\right)\right) \\
\omega_4 &= \omega\left(+\,+\,-\,-\right) = 2e^{\beta J - \beta J'}\cosh\left(\beta\left(J_1 + J_2 - J_3 - J_4\right)\right) \\
\omega_5 &= \omega\left(+\,-\,+\,+\right) = 2\cosh\left(\beta\left(J_1 - J_2 + J_3 + J_4\right)\right) \\
\omega_6 &= \omega\left(+\,+\,+\,-\right) = 2\cosh\left(\beta\left(J_1 + J_2 + J_3 - J_4\right)\right) \\
\omega_7 &= \omega\left(+\,+\,-\,+\right) = 2\cosh\left(\beta\left(J_1 + J_2 - J_3 + J_4\right)\right) \\
\omega_8 &= \omega\left(-\,+\,+\,+\right) = 2\cosh\left(\beta\left(-J_1 + J_2 + J_3 + J_4\right)\right).
\end{aligned}
\tag{4.6}
$$

There can only be eight possible arrangements of the four spin variables and so we can consider it to be an eight-vertex model with weights (4.5). This eight-vertex model was proved to satisfy the free fermion condition by Fan and Wu [FW70].

$$
\omega_1\omega_2 + \omega_3\omega_4 = \omega_5\omega_6 + \omega_7\omega_8
\tag{4.7}
$$

and as such is a free fermion model with the partition function:

$$
Z = \frac{1}{2}Z_l.
$$

Here the factor $1/2$ takes account of the two-to-one mapping of the spin and vertex configurations. The spontaneous magnetisation of a free fermion model was given by Baxter in [Bax86] and is

$$
\langle\sigma\rangle = \begin{cases} \left(1 - \Omega^{-2}\right)^{1/8}, & \Omega^{-2} \geq 1 \\ 0, & \Omega^{-2} \leq 1, \end{cases}
\tag{4.8}
$$

$$
\Omega^2 = 1 - \frac{\gamma_1\,\gamma_2\,\gamma_3\,\gamma_4}{16\omega_5\,\omega_6\,\omega_7\,\omega_8}.
\tag{4.9}
$$

where

$$
\begin{aligned}
\gamma_1 &= -\omega_1 + \omega_2 + \omega_3 + \omega_4 \\
\gamma_2 &= \omega_1 - \omega_2 + \omega_3 + \omega_4 \\
\gamma_3 &= \omega_1 + \omega_2 - \omega_3 + \omega_4 \\
\gamma_4 &= \omega_1 + \omega_2 + \omega_3 - \omega_4
\end{aligned}
\tag{4.10}
$$

The system exhibits an Ising transition at the critical point(s)

$$
\Omega^2 = 1
\tag{4.11}
$$

or equivalently,

$$
\omega_1 + \omega_2 + \omega_3 + \omega_4 = 2 \max\{\omega_1, \omega_2, \omega_3, \omega_4\}.
\tag{4.12}
$$

We can now go on to finding the spontaneous magnetisation for the other sublattice, $\mathcal{L}'$. While Lin and Wang presented a result for the symmetric case in 1988 [LW88], here we will quickly review the result for the general case given by Wu and Lin in [WL89]. Their approach uses the three site triplet interactions and again, as in section 3.3.2, we will simply present the result here. The τ-sublattice magnetisation is given by

$$
\langle \tau \rangle = \langle \sigma \rangle \left[A_{1234}(K)(F_+ + F_-) + A_{2341}(K)(F_+ - F_-) \right].
\tag{4.13}
$$

We can see that this equation has a similar form to (3.61), and can be seen to be a multiple of the σ-sublattice value. In (4.13),

$$
\begin{aligned}
A_{1234}(K) &= \frac{\sinh 2(\beta J_1 + \beta J_3)}{\sqrt{2 G_-(\beta J) \sinh 2\beta J_1 \sinh 2\beta J_3}} \\
A_{2341}(K) &= \frac{\sinh 2(\beta J_2 + \beta J_4)}{\sqrt{2 G_-(\beta J) \sinh 2\beta J_2 \sinh 2\beta J_4}}
\end{aligned}
$$

and

$$
G_-(\beta J) = \cosh 2(\beta J_1 + \beta J_3) + \cosh 2(\beta J_2 - \beta J_4).
$$

The calculation for F_+ and F_- is a little more involved. We start by calculating

$$F_\pm = \sqrt{\frac{A + 2\sqrt{BC}}{D + 2E\sqrt{B}}} \tag{4.14}$$

where

$$
\begin{aligned}
A &= 2\omega_5\omega_6\omega_7\omega_8\left(\omega_1^2 + \omega_2^2 + \omega_3^2\omega_4^2\right) - \left(\omega_1\omega_2 + \omega_3\omega_4\right)\left(\omega_1\omega_3 + \omega_2\omega_4\right)\left(\omega_1\omega_4 + \omega_2\omega_3\right) \\
B &= \omega_5\omega_6\omega_7\omega_8\left(\omega_5\omega_6\omega_7\omega_8 - \omega_1\omega_2\omega_3\omega_4\right) \\
C &= \left(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2\right)^2 - 4\left(\omega_5\omega_6 - \omega_7\omega_8\right)^2 \\
D &= \left(\omega_1^2 + \omega_2^2\right)\left(2\omega_5\omega_6\omega_7\omega_8 - \omega_1\omega_2\omega_3\omega_4\right) - \omega_5\omega_6\omega_7\omega_8\left(\omega_3^2 + \omega_4^2\right) \\
E &= \omega_1^2 - \omega_2^2.
\end{aligned}
$$

We can relate F_+ and F_- with the following formula, allowing us to get values for each variable,

$$F_+F_- = \frac{\omega_5\omega_6 - \omega_7\omega_8}{\omega_1\omega_2}. \tag{4.15}$$

We can compute the overall nearest neighbour magnetisation by taking the mean of the two sublattice magnetisations

$$M_0 = \frac{1}{2}(\langle\sigma\rangle + \langle\tau\rangle).$$

4.2.1 Classification of phases

At low temperatures the phase of the σ-sublattice can be classified based on the following energy value,

$$
\begin{aligned}
-E_1 &= J + J' + |J_1 + J_2 + J_3 + J_4| \\
-E_2 &= -J - J' + |J_1 - J_2 + J_3 - J_4| \\
-E_3 &= -J + J' + |J_1 - J_2 - J_3 + J_4| \\
-E_4 &= J - J' + |J_1 + J_2 - J_3 - J_4|
\end{aligned}
\tag{4.16}
$$

The sublattice is in a ferromagnetic phase when

$$E_1 < E_2,\ E_3,\ E_4;$$

is antiferromagnetic when

$$E_2 < E_1, \ E_3, \ E_4;$$

and finally is metamagnetic when

$$E_3 < E_1, \ E_2, \ E_4 \text{ or } E_4 < E_1, \ E_2, \ E_3.$$

As the temperature rises, depending on the relative strengths of the interactions J_r, the occurrence of phase change is signified by one or more of the following equations being realised

$$\omega_1 + \omega_2 + \omega_3 + \omega_4 = 2 \max \{\omega_1, \ \omega_2, \ \omega_3, \ \omega_4\}. \tag{4.17}$$

A re-entrant transition occurs if any one equation admits two solutions.

4.3 Theoretical analysis

In this section we will plot the theoretical predictions of Wu and Lin, and compare those against the critical temperatures from the results of Stephenson. To do this we will plot results from each of the possible systems, along with a plot of the γ from (4.9). Wu and Lin's predictions will be plotted against our simulation results later in Chapter 8. Here we are only concerned with identifying systems that need further investigation, due to conflict between the predictions of Vaks *et al.* and, Wu and Lin.

For our first system we will look at the isotropic ferromagnetic case. This is the simplest system as all the interactions are the same, $J_n = 100k_B$. The graph of this system is shown in Figure 4.3: We can see that the prediction of this system moves from having a non-zero average magnetic spin to a zero average magnetic spin. The system has this phase transition at the critical temperature, T_c around 400 Kelvin. This prediction agrees with the value obtained from α_1 in 4.3, and a solution to α_2 does not exist. Intuitively this is the sort of graph we would expect. Although for this system both theories agree, we will go on to examine the changes in the gamma terms. This will set the groundwork for our later analysis. This plot is shown in Figure 4.4. Here we can see that at low temperatures γ_1 is the only negative term, and that it equals zero at T_c and then moves to being positive and converges to the level of the other γ.
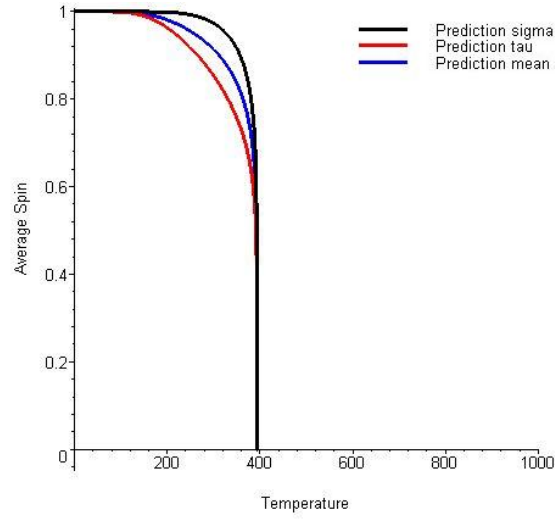
Figure 4.3: Theoretical predictions of Wu and Lin for a ferromagnetic system with interaction strengths $J_n = J = J' = 100k_B$. Here the prediction for the σ-sublattice (eqn. (4.8)) is shown in black, the prediction for the τ-sublattice (eqn. (4.13)) is shown in red and the prediction for the overall lattice is shown in blue. A phase transition can be seen just below 400 Kelvin.
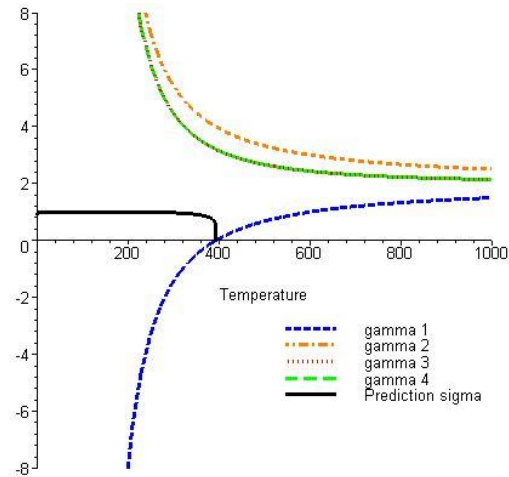
Figure 4.4: Graph of the separate γ terms from (4.10) and the σ-sublattice prediction for the ferromagnetic system with interactions strengths $J_n = J = J' = 100k_B$. Here γ_3 is overlaid on γ_4 as they are equal. Note that at low temperatures γ_1 is negative and the other terms are positive. At the critical temperature $\gamma_1 = 0$.

Now we will move on to looking at a system that starts in a metamagnetic phase. For this system we need to set the diagonals to be opposite polarities, that is $J = -J' = 100k_B$. Due to this the Vaks *et al.* result does not apply to this system. We also set the horizontal and vertical interactions to be $J_n = 10k_B$. The plots for this system are shown in Figure 4.5 As a metamagnetic system only has an average magnetic spin when
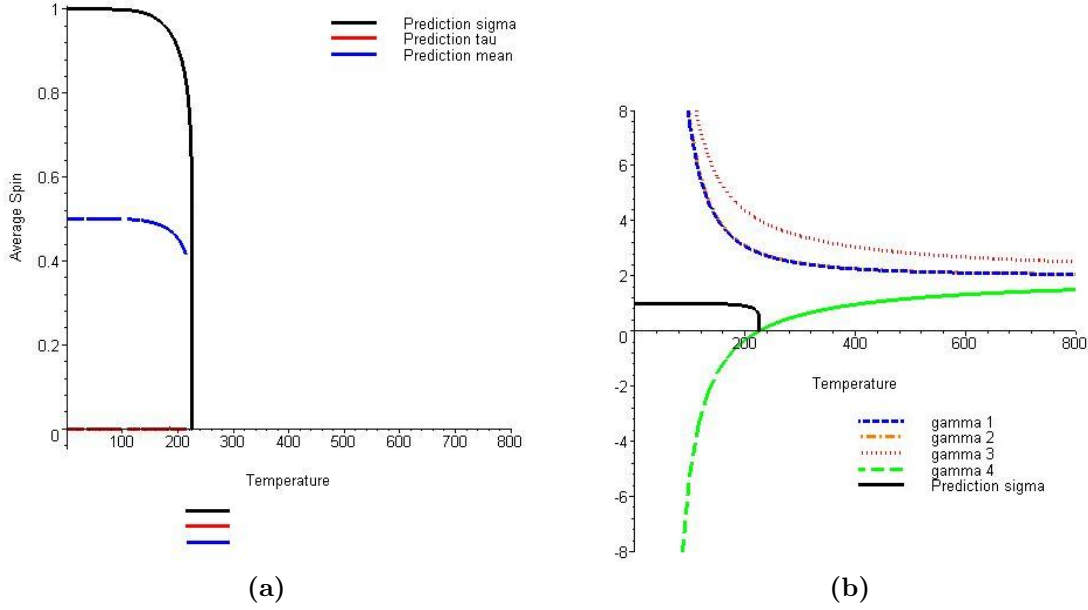
Figure 4.5: Results for a metamagnetic system with interaction strengths $J_n = 10k_B$, $J = -J' = 100k_B$. (a) shows the theoretical predictions of Wu and Lin. We see that the result for the τ-sublattice is zero across the temperature range while the σ-sublattice shows a transition at just above 200 Kelvin. (b) shows the values of the separate γ terms. Here we see that γ_4 is negative at low temperatures and crosses the x-axis at the transition temperature. In this graph γ_1 is overlaid on γ_2 as they are equal.

in the presence of an external magnetic field, we can see that something is incorrect here. From Figure 4.5b we can see that γ_4 is negative at low temperatures and behaves similarly to γ_1 in the ferromagnetic case. As

$$\gamma_4 = \omega_1 + \omega_2 + \omega_3 - \omega_4 < 0,$$

we note that $\omega_4 = \omega(++--)$ is the dominant term. As this term would suggest an antiferromagnetic phase we would intuitively expect $\langle\sigma\rangle = 0$.

The system we next consider is that on an anisotropic antiferromagnetic system. For this system we set the vertical and horizontal interactions to be $J_n = 100k_B$ and the diagonal interactions to be $J = J' = -100k_B$. As the system is antiferromagnetic

we expect it to have a zero average spin across the temperature range. The plots are shown in Figure 4.6 below. From Figure 4.6a we can see that the $\langle \tau \rangle$ prediction
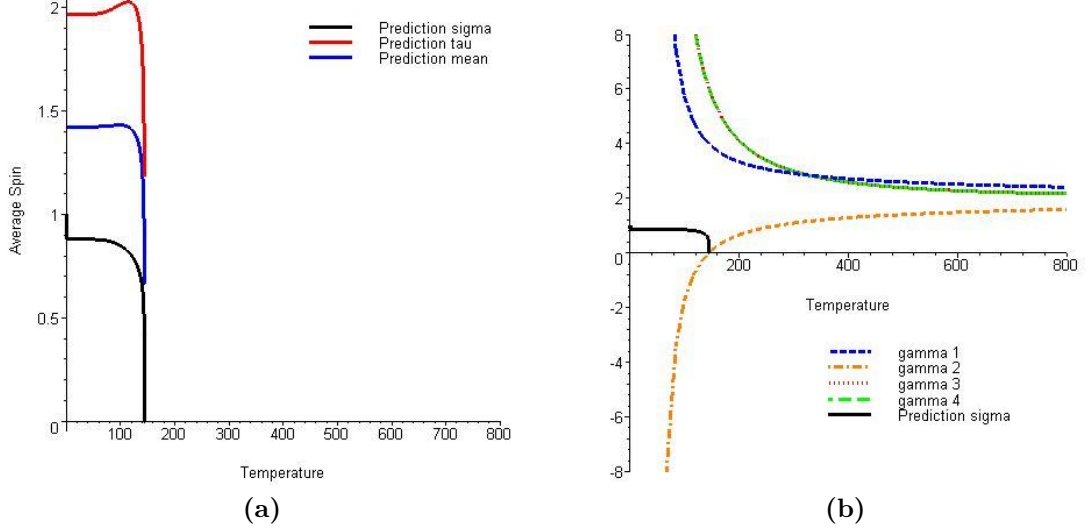
Figure 4.6: Results for the antiferromagnetic system with interactions $J_n = 100k_B$, $J = J' = -100k_B$. (a) shows the predictions from the results of Wu and Lin. Note that the graph for the τ-sublattice is above $+1$ which is physically impossible in our model. There is a transition below 200 Kelvin in all predictions. (b) shows the separate γ values for the system. Here γ_2 is identified as the dominant term for the system, and is negative while the predictions are non-zero. Again in this plot, γ_3 is overlaid on γ_4.

produces physically impossible results. In our system the values of the spin can only be ± 1 and so a result greater than one is impossible. As the value of $\langle \tau \rangle$ is determined as a multiple of the value of $\langle \sigma \rangle$ we should also look at this plot. We note that although our system is antiferromagnetic the prediction graph shows a non zero value for the magnetisation. As before, when we plot the individual γ in Figure 4.6b we see that this time it is γ_2 that is negative before the critical temperature. This implies that $\omega_2 = \omega(+ - +-)$ dominates, and so again should have average spin of zero. When we compare these predictions to those of Vaks *et al.* we can see that the critical temperature here is T_D, that is the disorder temperature where the system goes from being an ordered antiferromagnetic phase to a disordered antiferromagnetic phase.

As one of the motivations for studying the Union Jack lattice is the property of re-entrant phase transitions, we will now move on to looking at one such system. The system we will now study is the anisotropic ferromagnetic case. For this type of model we will look at the system where the square lattice has interactions with strength $J_n = 100k_B$ while the diagonals will have interaction strengths of $J = J' = -92k_B$.

When we calculate the possible critical temperatures from Vaks *et al.* , we see that all three are possible, and there exists a re-entrant phase transition. The graph of Wu and Lin's prediction is shown in Figure 4.7:

(a)   (b)
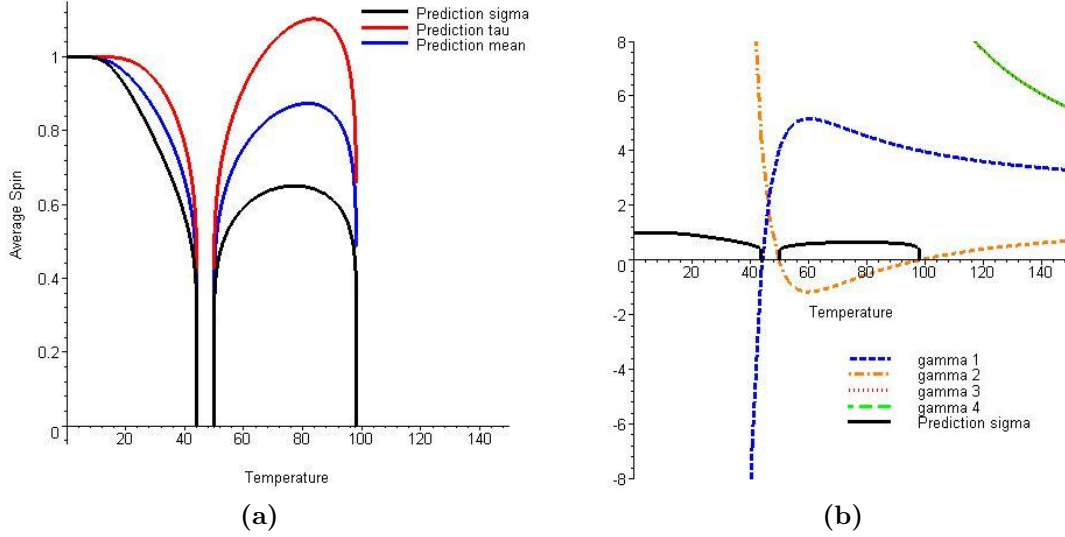
Figure 4.7: Results for the anisotropic ferromagnetic system with interaction strengths $J_n = 100k_B$, $J = J' = -92k_B$. This means that the horizontal and vertical interactions are equal. (a) shows the predictions from Wu and Lin for this system. Note after the first phase transition there is a second area of non-zero magnetisation. Here the prediction for the τ-sublattice again has physically impossible results given the conditions of our model. (b) shows graphs of the separate γ terms. Here we see that initially the γ_1 term is negative up to the first critical temperature. In the range of the second area of non-zero magnetisation we see that γ_2 is negative. As γ_3 is equal to γ_4 their plots are overlaid in this graph.

Looking at Figure 4.7a we see that in the set of curves between 45 and 100 Kelvin we have physically impossible results again for the $\langle \tau \rangle$ variable. Up to the first critical temperature we can see that the system behaves as we would expect a ferromagnetic system to, and then between the second and third critical temperatures we have the re-entrant phase. Examining Figure 4.7b we see that the first of these curves is determined when γ_1 is negative, suggesting a ferromagnetic system. During the re-entrant phase however, it is γ_2 which is negative, suggesting that its is an ordered antiferromagnetic region. In the paper of Vaks *et al.* [VLO66] the re-entrant phase transition is defined to be between a disordered antiferromagnetic phase and an ordered antiferromagnetic phase. From only looking at the average magnetisation of the system it is not possible to see the transition from a disordered phase to an ordered phase. However the critical temperatures here are consistent with those predicted from (4.3).

However, the results from some anisotropic ferromagnetic systems do have agreement between both theories. To show an example of this, we will look at the system where the interactions on the square lattice are $J_n = -100k_B$ and the diagonal interactions are $J = J' = 92k_B$. This is illustrated below in Figure 4.8.
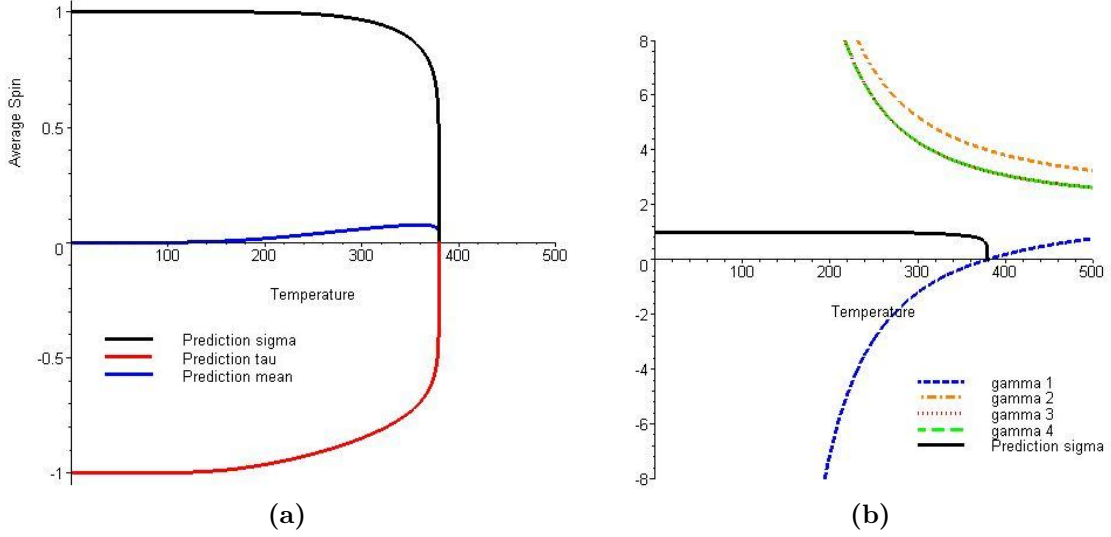
Figure 4.8: Results for the anisotropic ferromagnetic system with interactions strengths $J_n = -100k_B$, $J = J' = 92k_B$. These are the same interaction strengths as the last example but with the opposite sign. (a) shows the predictions of Wu and Lin for this system. Note that the overall system is antiferromagnetic with a zero average magnetisation but the sublattices have spins of different signs. As the temperature rises, due to the shape of the different curves, an overall magnetisation increases up to the critical temperature of the phase transition. (b) shows the graphs of the individual γ terms. Here we see that only γ_1 is negative in the temperature range, becoming positive at the critical temperature. In this graph γ_3 and γ_4 are equal and as such their plots are overlaid.

Looking at Figure 4.8a we can see that the curves are of opposite signs, but both display a ferromagnetic shape. Using the calculations from Vaks *et al.* we would expect the critical temperature to be at the value it is, and of the type it is. From examining Figure 4.8b we can see that it is only γ_1 which is negative at low temperatures. Equally as $\omega_1 = \omega(+ + ++) = \omega(- - --)$, the result for $\langle \tau \rangle$ is acceptable, even though the overall system is antiferromagnetic.

However there are some anisotropic systems, with non-uniform square lattice interactions where we see some interesting results. We shall now go on to look at a system where the horizontal interactions are $J_1 = J_3 = 100k_B/0.9^2$, the vertical interactions are $J_2 = J_4 = 100k_B/0.9$ and the diagonal interactions are $J = J' = 100k_B$. The graph

of this prediction is shown in Figure 4.9 below.

(a) Prediction results   **(b)** Separation of γ
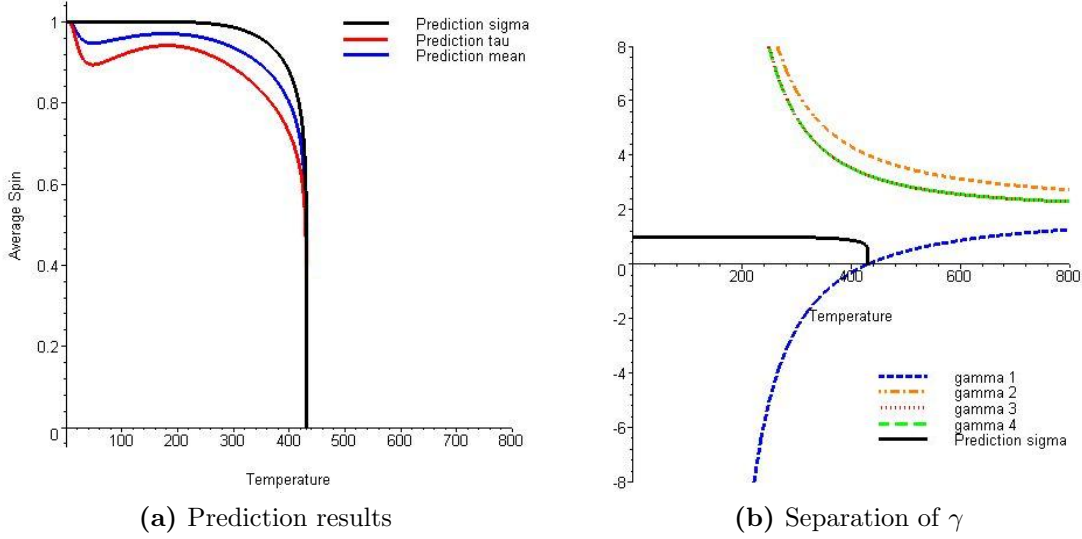
Figure 4.9: Results of the anisotropic ferromagnetic system with interactions horizontally of $J_1 = J_3 = 100k_B/0.9^2$, vertically of $J_2 = J_4 = 100k_B/0.9$ and diagonally of $J = J' = 100k_B$. (a) shows the predictions of Wu and Lin for the system. Note that the predictions are all physically plausible being less than or equal to $+1$. In the τ-sublattice prediction it can be seen that there is a little dip in the value before returning to the curve similar to the prediction function of the σ-sublattice. (b) shows the separate γ terms. As with previous ferromagnetic systems γ_1 is the dominant term, and the graphs of γ_3 and γ_4 are overlaid.

As we can see from Figure 4.9a, the curves for τ and subsequently the mean predictions deviate from the shape of the σ curve at temperatures below 200 Kelvin. Above this temperature, the graphs show the expected ferromagnetic curves. As there are no values over $+1$ this is physically plausible. When we look at the separated γ in Figure 4.9b the only term that crosses the x-axis is γ_1. This again suggests a purely ferromagnetic system, and shows the critical temperature where we would expect. We now look at a similar system where we swap around the interaction strengths of the vertical and horizontal interactions. The graph of this system is shown in Figure 4.10.

From Figure 4.10a we can see that again, the τ and so the mean magnetic spin results are physically impossible given our system. However we can see from this figure and Figure 4.10b that the prediction for σ remains the same. Intuitively we would expect this, as we have only rotated the system from Figure 4.9 by 90 degrees. Again we see that the predictions become more like we would expect above 200 Kelvin.
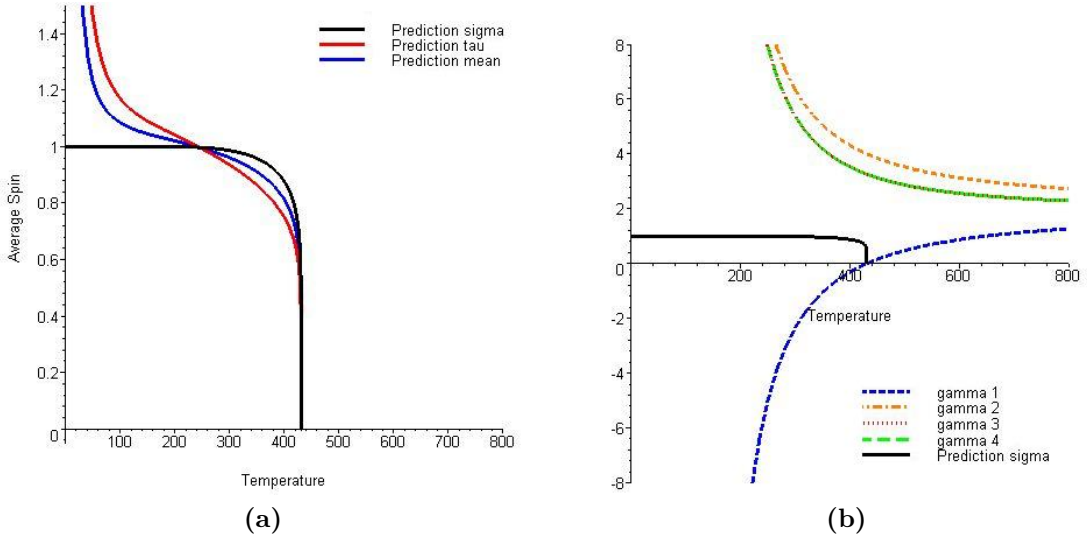
Figure 4.10: Results for the anisotropic ferromagnetic system with interaction strengths $J_1 = J_3 = 100k_B/0.9$, $J_2 = J_4 = 100k_B/0.9^2$ and $J = J' = 100k_B$. Note these interactions are those of Figure 4.9 rotated by 90 degrees. (a) shows the predictions of Wu and Lin for the system. Here we see that the prediction for the τ-sublattice is physically impossible in our model below around 250 Kelvin. (b) shows the graphs of the separate γ terms. Note that the result is identical to Figure 4.9b.

4.4   Summary

We have seen in this section that there are a number of systems for which the predicted results from Vaks *et al.* and Wu and Lin disagree. Equally we note that the Wu and Lin prediction only works when either $\gamma_1 < 0$ or $\gamma_1 \, \gamma_2 \, \gamma_3 \, \gamma_4 > 0$. These added conditions were not stated in either Wu and Lin's 1987 [WL87] or 1989 [WL89] papers. This suggests that these results need to be investigated further with other methods, such as mean field theory in Chapter 5 and numerical simulations in Chapter 8.

Mean Field Theory

In this chapter we will briefly discuss alternative ways of investigating the Ising model through the use of approximations. While exact solutions are useful for investigating systems under special conditions, such as in the absence of a magnetic field, approximations allow us to look at systems where an exact solution may not be possible. After this discussion we will look at one particular method of approximation, mean field theory, in more detail. Initially we will look at how mean field theory is defined for the triangular lattice model [Bax89], which is a specific case of the mean field theory method for the isotropic lattice. After this we will develop our own equations for the Union Jack lattice. At the end of the chapter we will the perform some numerical analysis to investigate how well the approximation models our systems.

5.1 Approximations

Mean field theory is an older approach to the subject of phase transitions and generally give us a qualitative description of the phenomena of interest. There are many different approaches that can be use with mean field theory, but common to all is the identification of an order parameter. For this section we will be using the Curie-Weiss model [PB94]. In this method we approximate a system where nearest neighbours interact with each other, to a system where the particles do not interact with each other but instead with the mean spin of the entire lattice. This means that the approximation is affected less by the dimensionality or lattice configuration of the system and so we are able to avoid the issues discussed in Section 1.2 (work by [Bar82, Ist00]). This will allow us to investigate systems with external magnetic field in two dimensions. Alternatively, a similar approach where the free energy is expressed, then minimised

with respect to the order parameter, can be used. An example of this free energy minimisation approach is the Bragg-Williams approximation [BW34, Yew70].

These methods can be improved to gain numerical values for the properties of the Ising model, by such approximations as those presented by Bethe in [Bet35] and its extension by Guggenheim [FG40]. However, the asymptotic critical behaviour of mean field theories is always the same. The most serious fault of mean field theories lies in the neglect of long-range fluctuations of the order parameter. The importance of this omission depends very much on the dimensionality of the problem. In problems involving one and two-dimensional systems, the results predicted by mean field theory are often quantitatively wrong, although they do give a qualitative idea of the behaviour.

5.2   The triangular lattice model

First we will look at the development of mean field theory on the triangular lattice, which we will use as a basis for our development on the Union Jack lattice. This is a well studied area, with many textbooks on the subject, including [LL80], [KIUH65] (summarised in [PB94]) and [Bax89].

Let us consider the triangular lattice Ising model in an external magnetic field, B. The Hamiltonian of such a system is given by

$$H = -J \sum_{\langle i,j \rangle}^{N} \sigma_{i,j}\sigma_{i+1,j} + \sigma_{i,j}\sigma_{i,j+1} + \sigma_{i,j}\sigma_{i+1,j+1} - B \sum_{i,j=1}^{N} \sigma_{i,j}. \tag{5.1}$$

Now we focus on one site and consider its expectation value, which at a certain temperature has value m,

$$\langle \sigma_i \rangle = m \tag{5.2}$$

for all i. This m is referred to as order parameter of the system. For the next step, we will focus on a particular spin of the lattice σ_0. As in our final model all spins will be interacting with the parameter m. We can first develop the method for one site with j nearest neighbours and then extend it to the entire lattice. For this we take the local

version of the Hamiltonian in (5.1), which is

$$
\begin{aligned}
h\left(\sigma_0\right) &= -\sigma_0\left(J\sum_j \sigma_j + B\right) \\
&= -\sigma_0\left(qJm + B\right) - J\sigma_0\sum_j\left(\sigma_j - m\right)
\end{aligned}
\tag{5.3}
$$

where q is half the number of nearest neighbours of site 0. This definition of q is such that the duplication of counting of nearest neighbour spins can be eliminated. If we disregard the second term in this equation we obtain a non-interacting system. In a non-interacting system each spin is an effective magnetic field composed of the applied field and an average exchange field due to the neighbours. This means that it will work in any number of dimensions, since the lattice configuration only effects the value of q. For example $q = 2$ for a square lattice, $q = 3$ for both the triangular and simple cubic lattice. The magnetisation has to be determined self-consistently from the condition

$$
m = \langle\sigma_0\rangle = \langle\sigma_j\rangle, \ \ j = 1, 2, \ldots N.
$$

Extending the local Hamiltonian for an non-interacting system to all sites in the matrix we obtain the following equation,

$$
H = -Jqm\sum_{i=1}^{N}\sigma_i - B\sum_{i=1}^{N}\sigma_i.
\tag{5.4}
$$

As before, the partition function can be found from the Hamiltonian

$$
\begin{aligned}
Z_N &= \sum_\sigma \exp\left[\left(\beta Jqm + \beta B\right)\sum_{i=1}^{N}\sigma_i\right] \\
&= 2\cosh\left(\beta Jqm + \beta B\right)Z_{N-1} \\
\text{where } Z_{N-1} &= \sum_{\sigma'}\exp\left[\left(\beta Jqm + \beta B\right)\sum_{i=1}^{N-1}\sigma_i\right]
\end{aligned}
\tag{5.5}
$$

and σ' denotes a configuration over $N - 1$ sites, $\beta = 1/(k_B T)$.

Now we return to looking at the specific case for σ_0

$$
\begin{aligned}
m & = \langle \sigma_0 \rangle \\
& = Z_N^{-1} \sum_\sigma \sigma_0 \exp\left[(\beta Jqm + \beta B)\sum_{i=1}^{N}\sigma_i\right] \\
& = Z_N^{-1} 2\sinh\left(\beta Jqm + \beta B\right)\sum_{\sigma'}\exp\left[(\beta Jqm + \beta B)\sum_{i=1}^{N-1}\sigma_i\right] \\
& = Z_N^{-1}\sinh\left(\beta Jqm + \beta B\right)Z_{N-1} \\
& = \tanh\left(\beta Jqm + \beta B\right).
\end{aligned}
\tag{5.6}
$$

To find m, the above equation must be solved numerically for B and T. This equation is invariant under the map $B \to -B$, $m \to -m$

$$
\begin{aligned}
-m & = \tanh\left(-\beta Jqm + -\beta B\right) \\
& = -\tanh(-\left(\beta Jqm + \beta B\right)) \\
m & = \tanh\left(\beta Jqm + \beta B\right).
\end{aligned}
\tag{5.7}
$$

It can be seen from this that for a nonzero external field there is at least one solution and sometimes three. For a system with no external field there is always the solution $m = 0$, and two further solutions may exist at $\pm m_0$ where m_0 signifies the spontaneous magnetisation. As $T \to 0$, $\tanh\left(\beta qJm\right) \to \pm 1$ for $m \neq 0$ and $m_0 \to \pm 1$. As T approaches the critical temperature T_c from $T < T_c$, $|m_0(T)|$ decreases and we can obtain its asymptotic dependence from a low order Taylor series expansion of the hyperbolic tangent. That is,

$$
m_0 = \beta qJm_0 - \frac{1}{3}\left(\beta qJ\right)^3 m_0^3 + \cdots
$$

or

$$
m_0(T) = \pm\sqrt{3}\left(\frac{T}{T_c}\right)^{3/2}\left(\frac{T_c}{T} - 1\right)^{1/2}.
\tag{5.8}
$$

From this we can see that the order parameter m approaches zero in a singular fashion

as T approaches T_c, vanishing asymptotically as

$$m_0(T) \propto \left(\frac{T_c}{T} - 1\right)^{1/2}. \tag{5.9}$$

It should be noted at this stage that when we compare this with (3.56) that the power is $\frac{1}{2}$ rather than $\frac{1}{8}$, so will only give a qualitative result, as will be seen later in Figure 5.3.

5.3 Extending to the Union Jack

As we have stated before in Chapter 4, the Union Jack lattice can be thought of as two square lattices superimposed on each other. This means that we can develop the mean field approximation for this lattice in one of two ways, both of which will be shown in this section. First, in the most simple development, we can consider these sublattices independently of each other and thus develop a set of partially uncoupled equations. In a slightly more complex development, we can approximate the complete lattice with a set of coupled equations. That is, equations where there is a dependence on both sublattices. In the next section we will perform some analysis of the quality of the approximation of both systems.

5.3.1 Partially uncoupled equations

The partially uncoupled equations can be quickly developed from the equation for a triangular lattice, that is the sublattices have sites with the same number of nearest neighbours. As in Chapter 4 we denote the sites with four nearest neighbours as τ_k and the sites with eight nearest neighbours as σ_j. We now simplify the system further by taking the interaction strengths along the vertical and horizontal bonds as J, and along the diagonals as K. We also include an applied external magnetic field of B. Using the above theory for triangular lattices, we can construct equations for each type of site:

$$
\begin{aligned}
\langle \sigma \rangle &= \tanh\left[\beta\left(2K\langle\sigma\rangle + B\right)\right], \\
\langle \tau \rangle &= \tanh\left[\beta\left(4J\langle\sigma\rangle + B\right)\right].
\end{aligned}
\tag{5.10}
$$

In these equations we can see that the number of nearest neighbours is perhaps not what we would expect from the previous section. In the equation for $\langle\sigma\rangle$, we are only concerned with interactions between one σ and another σ, so to avoid duplication we have to divide by two. Meanwhile in the equation for $\langle\tau\rangle$, we are concerned with the number of interactions between a τ and a σ, so we do not need to worry about duplication as there are no $\tau - \tau$ interactions. From this point forward we will refer to equations (5.10) as the *partially uncoupled equations*, since both equations are dependent on $\langle\sigma\rangle$ and so only the equation for $\langle\sigma\rangle$ is uncoupled.

5.3.2   Coupled equations

While the partially uncoupled equations can be useful, to make the equations uncoupled we purposefully ignored the interactions with the $\langle\tau\rangle$ in the equation for $\langle\sigma\rangle$. Now we will develop a system of equations where we consider all the interactions on the σ, which will be referred to as the *coupled equations*. Considering the lattice as before, we can rewrite the Hamiltonian for the system as

$$H = -J \sum_{\langle j,k\rangle} \sigma_j \tau_k - K \sum_{\langle j,l\rangle} \sigma_j \sigma_l - B \sum_j \sigma_j - B \sum_k \tau_k.$$

Again we focus on the expectation values of the sites and put them into our Hamiltonian to get

$$H = -2J \left(\sum_k \langle\sigma\rangle \tau_k + \sum_j \langle\tau\rangle \sigma_j \right) - 2K \sum_j \langle\sigma\rangle \sigma_j + B \sum_j \sigma_j + B \sum_k \sigma_k.$$

The partition function is then

$$Z_N = \sum_{\sigma_j,\tau_k=\pm1} \exp\left[\beta\left(2K\langle\sigma\rangle + 2J\langle\tau\rangle + B\right) \sum_j \sigma_j + \beta\left(2J\langle\sigma\rangle + B\right) \sum_k \tau_k \right].$$

The partition function is defined as the sum over all possible values of σ_j and τ_k, and there are $N = L^2$ sites in total. Let us pick two such spins, say σ_P and τ_Q, and perform

that part of the sum involving these sites.

$$\sum_{\sigma_P=\pm 1} \exp\left[\beta\left(2K\left\langle\sigma\right\rangle + 2J\left\langle\tau\right\rangle + B\right)\sigma_P\right] = \exp\left[\beta\left(2K\left\langle\sigma\right\rangle + 2J\left\langle\tau\right\rangle + B\right)\right]$$
$$+ \exp\left[-\beta\left(2K\left\langle\sigma\right\rangle + 2J\left\langle\tau\right\rangle + B\right)\right]$$
$$= 2\cosh\left[\beta\left(2K\left\langle\sigma\right\rangle + 2J\left\langle\tau\right\rangle + B\right)\right].$$

Likewise

$$\sum_{\tau_Q=\pm 1} \exp\left[\beta\left(2J\left\langle\sigma\right\rangle + B\right)\tau_Q\right] = 2\cosh\left[\beta\left(2J\left\langle\sigma\right\rangle + B\right)\right].$$

If Z_{N-2} is the partition function for the same system but with $N-2$ lattice sites, then

$$Z_N = 4\cosh\left[\beta\left(2K\left\langle\sigma\right\rangle + 2J\left\langle\tau\right\rangle + B\right)\right]\cosh\left[\beta\left(2J\left\langle\sigma\right\rangle + B\right)\right]Z_{N-2}.$$

Now we compute the magnetisation for σ_P and τ_Q:

$$\left\langle\sigma_P\right\rangle = Z_N^{-1}\sum_{\sigma_j,\tau_k=\pm 1}\sigma_P\exp\left[\beta\left(2K\left\langle\sigma\right\rangle + 2J\left\langle\tau\right\rangle + B\right)\sum_j\sigma_j + \beta\left(2J\left\langle\sigma\right\rangle + B\right)\sum_k\tau_k\right],$$

$$\left\langle\tau_Q\right\rangle = Z_N^{-1}\sum_{\sigma_j,\tau_k=\pm 1}\tau_Q\exp\left[\beta\left(2K\left\langle\sigma\right\rangle + 2J\left\langle\tau\right\rangle + B\right)\sum_j\sigma_j + \beta\left(2J\left\langle\sigma\right\rangle + B\right)\sum_k\tau_k\right].$$

Now we use the fact that

$$\sum_{\sigma_P=\pm 1}\sigma_P\exp\left[\beta\left(2K\left\langle\sigma\right\rangle + 2J\left\langle\tau\right\rangle + B\right)\sigma_P\right] = 2\sinh\left[\beta\left(2K\left\langle\sigma\right\rangle + 2J\left\langle\tau\right\rangle + B\right)\right],$$

$$\sum_{\tau_Q=\pm 1}\tau_Q\exp\left[\beta\left(2J\left\langle\sigma\right\rangle + B\right)\tau_Q\right] = 2\sinh\left[\beta\left(2J\left\langle\sigma\right\rangle + B\right)\right]$$

to deduce

$$
\begin{aligned}
\langle \sigma_P \rangle &= Z_N^{-1} \sum_{\sigma_j, \tau_k = \pm 1} \sigma_P \exp \left[\beta \left(2K \langle \sigma \rangle + 2J \langle \tau \rangle + B \right) \sum_j \sigma_j + \beta \left(2J \langle \sigma \rangle + B \right) \sum_k \tau_k \right] \\
&= \frac{4}{Z_N} \sinh \left[\beta \left(2K \langle \sigma \rangle + 2J \langle \tau \rangle + B \right) \right] \cosh \left[\beta \left(2J \langle \sigma \rangle + B \right) \right] Z_{N-2} \\
&= \tanh \left[\beta \left(2K \langle \sigma \rangle + 2J \langle \tau \rangle + B \right) \right], \\
\langle \tau_Q \rangle &= Z_N^{-1} \sum_{\sigma_j, \tau_k = \pm 1} \tau_Q \exp \left[\beta \left(2K \langle \sigma \rangle + 2J \langle \tau \rangle + B \right) \sum_j \sigma_j + \beta \left(2J \langle \sigma \rangle + B \right) \sum_k \tau_k \right] \\
&= \frac{4}{Z_N} \sinh \left[\beta \left(2J \langle \sigma \rangle + B \right) \right] Z_{N-2} \cosh \left[\beta \left(2K \langle \sigma \rangle + 2J \langle \tau \rangle + B \right) \right] \\
&= \tanh \left[\beta \left(2J \langle \sigma \rangle + B \right) \right].
\end{aligned}
$$

Since we have $\langle \sigma_P \rangle = \langle \sigma \rangle$ for all P and $\langle \tau_Q \rangle = \langle \tau \rangle$ for all Q we need to analyse the equations

$$
\begin{aligned}
\langle \sigma \rangle &= \tanh \left[\beta \left(2K \langle \sigma \rangle + 2J \langle \tau \rangle + B \right) \right], \\
\langle \tau \rangle &= \tanh \left[\beta \left(2J \langle \sigma \rangle + B \right) \right].
\end{aligned}
\tag{5.11}
$$

5.3.3   Wu and Lin adapted

To test our mean field approximations against the known results of Wu and Lin [WL87, WL89] we will have to rewrite the conditions that they use to classify the type of system. To quickly review, for a ferromagnetic system the following conditions hold:

$$
E_1 < E_2, E_3, E_4;
\tag{5.12}
$$

for antiferromagnetic systems:

$$
E_2 < E_1, E_3, E_4;
\tag{5.13}
$$

and for the metamagnetic system:

$$
\begin{aligned}
E_3 &< E_1, E_2, E_4 \\
\text{or } E_4 &< E_1, E_2, E_3;
\end{aligned}
\tag{5.14}
$$

where

$$
\begin{aligned}
-E_1 &= J + J' + |J_1 + J_2 + J_3 + J_4|, \\
-E_2 &= -J - J' + |J_1 - J_2 + J_3 - J_4|, \\
-E_3 &= -J + J' + |J_1 - J_2 - J_3 + J_4|, \\
-E_4 &= J - J' + |J_1 + J_2 - J_3 - J_4|.
\end{aligned}
\tag{5.15}
$$

Now we rewrite (5.15) with the horizontal and vertical interactions being set to J and the diagonal interactions set to K, as before. This produces the following equations:

$$
\begin{aligned}
-E_1 &= K + K + |J + J + J + J| = 2K + 4|J|, \\
-E_2 &= -K - K + |J - J + J - J| = -2K, \\
-E_3 &= -K + K + |J - J - J + J| = 0, \\
-E_4 &= K - K + |J + J - J - J| = 0.
\end{aligned}
\tag{5.16}
$$

Using these equations we can rewrite our conditions for the phase of system as:
to be ferromagnetic:

$$
K + |J| > 0;
\tag{5.17}
$$

to be antiferromagnetic:

$$
K + |J| < 0 \text{ so } K < -|J|;
\tag{5.18}
$$

and finally to be metamagnetic

$$
K + 2|J| < 0 \text{ and } K > 0.
\tag{5.19}
$$

However $|J|$ is always positive, so the conditions for metamagnetic systems cannot be satisfied. This is due to our simplification to two variables, and means that mean field theory will not be able to model metamagnetic systems. At $K = 0$ it can be seen that the Union Jack lattice reduces to the square lattice.

5.4   Analysis of the magnetisation

Having now developed the mean field approximation for the Union Jack lattice, we will now analyse the equations (5.10) and (5.11). We will look at the two types of equations separately, looking at such quantities as the predicted critical temperatures. Then we will go on to look at the graphs of the functions at a constant temperature. As the equations for $\langle \tau \rangle$ rely on $\langle \sigma \rangle$ we will not be discussing them in this section, as they will follow those of $\langle \sigma \rangle$. Equally as the partially uncoupled equations are adapted forms of those from the square lattice, the analysis will be similar and so not included here (see [Bax89]).

5.4.1   Partially uncoupled equations

As the equation for $\langle \sigma \rangle$ in (5.10) only depends on itself no further rearrangement is required. For this analysis, and simplicity, we consider a system with no external magnetic fields applied, that is $B = 0$. For any temperature T, the corresponding value of $\langle \sigma \rangle$ is determined by the point of intersection of the curves $y = x$ and $y = \tanh\left[4\beta K \langle \sigma \rangle\right]$. The slope of the latter curve varies from the initial value of $4\beta K$ at the origin and asymptotically approaches the final value of zero. This can be seen graphically in Figure 5.1 below. It can be seen that there exists a critical temperature above which will have no spontaneous magnetisation. That is the magnetisation will be zero. It follows that an intersection of the two curves (other than at the origin) can only eventuate if

$$4\beta K > 1 \quad \text{or} \quad T < T_c = \frac{4K}{k_B}, \tag{5.20}$$

where we call T_c the critical temperature. At temperatures below T_c the model acquires a non-zero spontaneous magnetisation but at temperature $T > T_c$ no spontaneous magnetisation is possible.

5.4.2   Coupled equations

Now we move to the coupled equations and perform a similar analysis. However, we first need to rearrange the equations to make the equation for σ dependent only on the value of σ. We can do this easily as the equation for τ only depends on σ so we can
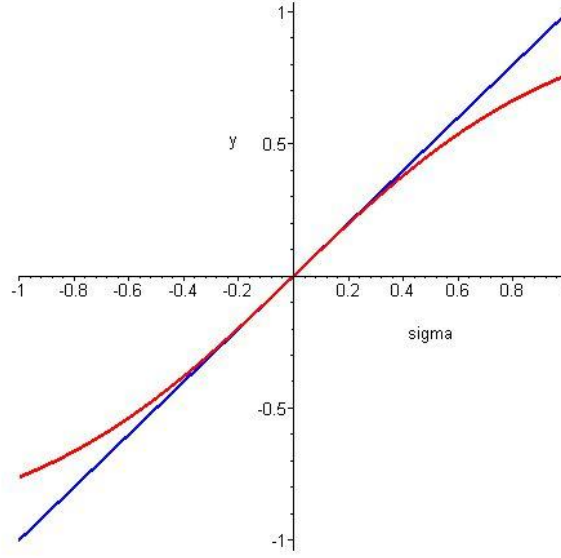
Figure 5.1: Graph of the partially uncoupled mean field prediction on the Union Jack lattice for σ (from eqn. (5.10)), shown in red, against the graph of $y = x$, shown in blue. The intersection of the two lines shows the value of $\langle \sigma \rangle$ at temperature T. Here $T = T_c$ and so the graphs intersect at exactly zero.

simply substitute it in as shown below:

$$
\begin{aligned}
\langle \sigma \rangle &= \tanh\left[\beta\left(2K\langle \sigma \rangle + 2J\langle \tau \rangle + B\right)\right], \\
\langle \tau \rangle &= \tanh\left[\beta\left(2J\langle \sigma \rangle + B\right)\right], \\
\text{thus } \langle \sigma \rangle &= \tanh\left[\beta\left(2K\langle \sigma \rangle + 2J\tanh\left[\beta\left(2J\langle \sigma \rangle + B\right)\right] + B\right)\right].
\end{aligned}
\tag{5.21}
$$

While this substitution has reduced the number of variables, it has also made the analysis more complicated. However at the origin the equation still behaves like the linearised version. We can find a critical temperature equivalently by solving

$$
2\beta K + 4\beta^2 J^2 > 1,
$$

or equivalently by solving

$$
4\beta^2 J^2 + 2\beta K - 1 = 0.
\tag{5.22}
$$

for β. We put the coefficients into the solution for a quadratic equation

$$
\begin{aligned}
\beta &= \frac{-2K \pm \sqrt{4K^2 + 16J^2}}{8J^2} \\
&= \frac{-K \pm \sqrt{K^2 + 4J^2}}{4J^2}.
\end{aligned}
\tag{5.23}
$$

The critical temperature can be found with the following formula

$$
T_c = \frac{4J^2}{k_B \left(-K \pm \sqrt{K^2 + 4J^2}\right)}.
\tag{5.24}
$$

This formula produces two roots, and we need to perform some further analysis to find which root we should take. We can rewrite (5.23) as a Taylor series expansion for small $|J|$ where the first term will be

$$
\beta \sim \frac{-K \pm |K| \left(1 + \frac{2J^2}{K^2}\right)}{4J^2}.
\tag{5.25}
$$

So for $K > 0$ we first take the positive sign, getting

$$
\beta \sim \frac{-K + K \left(1 + \frac{2J^2}{K^2}\right)}{4J^2} \sim \frac{1}{2K};
\tag{5.26}
$$

and if we take the negative sign we get

$$
\beta \sim \frac{-K - K \left(1 + \frac{2J^2}{K^2}\right)}{4J^2} \sim \frac{-K}{2J^2} - \frac{1}{2K}.
\tag{5.27}
$$

Now we look when $K < 0$, and again first take the positive sign:

$$
\beta \sim \frac{-K - K \left(1 + \frac{2J^2}{K^2}\right)}{4J^2} \sim \frac{-K}{2J^2} - \frac{1}{2K};
\tag{5.28}
$$

and finally taking the negative sign:

$$
\beta \sim \frac{-K + K \left(1 + \frac{2J^2}{K^2}\right)}{4J^2} \sim \frac{1}{2K}.
\tag{5.29}
$$

For ferromagnetic systems, when $K > -|J|$, we require a positive critical temperature. So we take the positive sign, for $K > 0$ using (5.26) and for $K < 0$ using (5.28). However, as J approaches zero, the σ sublattice becomes a square lattice with the critical temperature defined by $T_c = 2K/k_B$. For antiferromagnetic systems we require a zero or negative temperature, so we can see for consistency as J approaches zero, we must take the negative sign for $K < -|J|$ (5.29). Here K will always be negative, so we only use one equation.

Again we plot the graph of this function at the critical temperature, shown below in Figure 5.2. We can see at the critical point of the system the function follows the
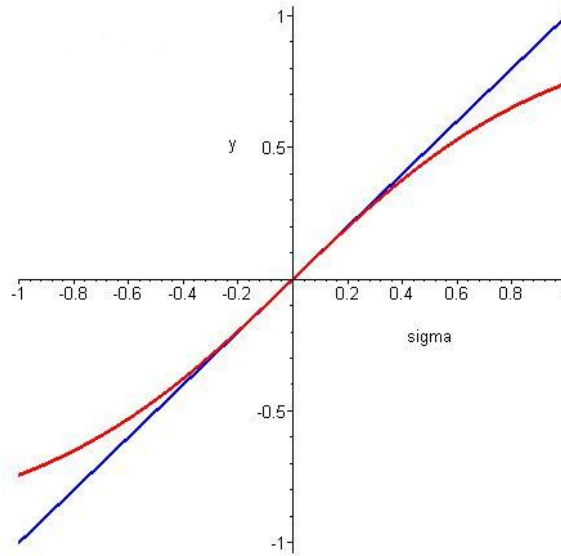
Figure 5.2: The graph of the coupled mean field prediction on the Union Jack lattice for σ (from eqn. (5.21)) is shown in red against the graph of $y = x$ shown in blue. For this prediction we set the temperature of the system to be again T_c. Again the intersection of the two lines shows the final value of σ, here at zero.

line near the origin. As we have seen before with the sites with four bonds, as the system passes the critical temperature it falls below this line.

5.5 Numerical results

Having developed our mean field approximations we can test them against our known results. In this section we will qualitatively compare results from a numerical simulation of the mean field approximation (program code given in Appendix C) to the known results of Wu and Lin [WL87]. While we do not expect the quantitative correlation

to be very close, qualitatively the approximation should bring a general idea of the character of the magnetisation. We will start by looking at the ferromagnetic square lattice and then we will move on to looking at a similar system on the Union Jack lattice.

5.5.1 Triangular ferromagnetic lattice

First we will look at a triangular lattice where the interactions have strength one hundred times the Boltzmann constant and there is no external magnetic field. In Figure 5.3 our data points are plotted against the known result. As we can see, the
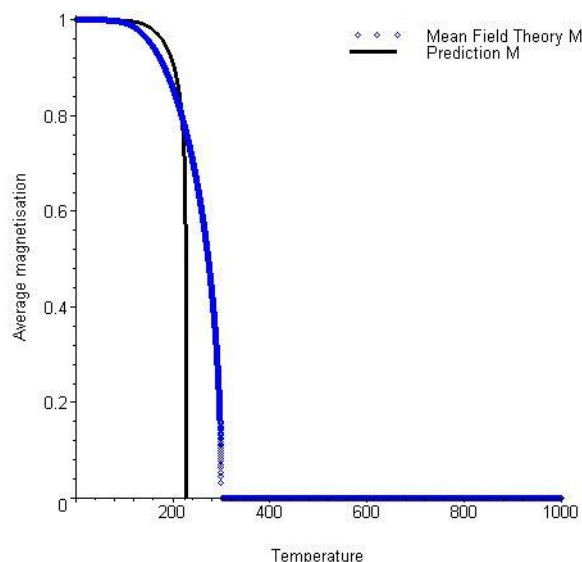
Figure 5.3: Graphs for the triangular lattice Ising model for a ferromagnetic system with interaction strength $J = 100k_B$. Here the solid line shows the prediction of Stephenson for nearest neighbour interactions (eqn. (3.57)). The data points show the numerical results from the mean field simulation of the system. Note that the critical temperature for the mean field curve is higher than the prediction temperature. However qualitatively the curves are similar.

mean field result is approximately the same shape, and shows a phase transition along with a critical temperature, of 300 K. That is a critical temperature higher than the known result. However the mean field result starts to move away from the positive ground state earlier than the analytical result and so has a slightly less steep curve. As a qualitative approximation it is not bad, though quantitatively it is quite far from the mark.

5.5.2 Union Jack ferromagnetic lattice

Next we move on to the Union Jack lattice. In general we have seen a good correlation between the theories of Stephenson [Ste70], Wu and Lin [WL87], and our intuitive expectation in Section 4.3. So as our first example of the results we get from our mean field approximation, we will look at the isotropic ferromagnetic case with interaction strengths of one hundred times the Boltzmann constant. The graph of the coupled equations versus the Wu and Lin's result is shown in Figure 5.4 below. Again we have
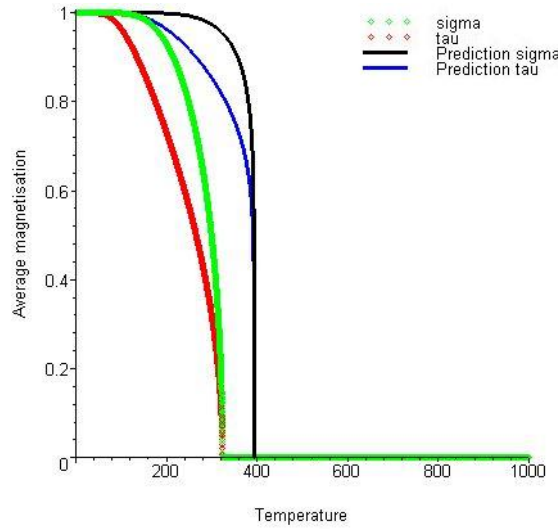
Figure 5.4: Graphs for the isotropic ferromagnetic system on the Union Jack lattice with interaction strengths $J = K = 100k_B$. The theoretical predictions of Wu and Lin are shown by the solid lines and the numerical results from the coupled mean field equations are shown by the points. Note that the curves are qualitatively the same though quantitatively the critical temperature of the mean field results is lower than the predictions.

a good qualitative correlation between the two sets of results, although quantitatively the approximation is slightly under the known result. When we compare to Figure 5.3 we see that it is of about the same quality as the triangular lattice predictions.

When we look at the partially uncoupled equations we obtain Figure 5.5 below. As we can see the correlations of this mean field theory approximation with the known results is quantitatively not good. Equally the values for τ are larger than those of the σ, which is in the opposite order to the known result. It is a very rough prediction and even when looking at it qualitatively we can see that the curves of the magnetism for the two sublattices are different. In general it would be of more use to use the coupled equations as those results are closer to the known theory, so for the rest of the section
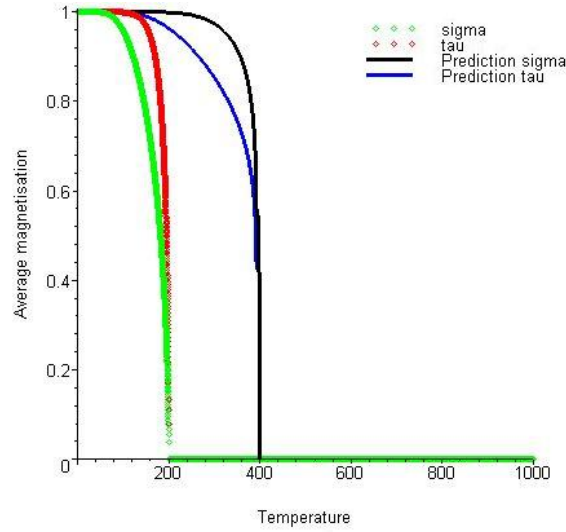
Figure 5.5: Graphs for the isotropic ferromagnetic system on the Union Jack lattice with interaction strengths $J = K = 100k_B$. Again the theoretical predictions of Wu and Lin are shown by the solid lines. The data points show the numerical results from the mean field partially uncoupled equations. Note that the order of the curves for the mean field predictions is opposite to the theoretical predictions. Also the critical temperature of the mean field is much lower than the theoretical prediction.

we will concentrate on those equations in (5.11).

Here we see that the results from the couple equations are more accurate than those from the uncoupled equations on isotropic systems. Next we shall look a anisotropic system. Of course this allows us to look at a system with a re-entrant phase transition. So we will look at the system with horizontal and vertical interactions of $J_n = 100k_B$ and diagonal interactions of $J = J' = 92k_B$. The graph we obtain is shown in Figure 5.6. The result from the mean field has a poor correlation with the prediction of Wu and Lin. Qualitatively it can seen that while the sublattice magnetisations are ordered in the same way as the predictions, but do not see the re-entrant phase transition at all. Interestingly the mean field result here quantitatively is above the critical temperature. Also we can see in the graph for the $\langle \sigma \rangle$ that there is a deviation from a smooth curve.

5.6   Summary

In this section, we have shown that the mean field approximation is a powerful tool for qualitative prediction of a given system in some instances. It allows us to study systems outside the range of the exact solutions as it is not limited by dimensionality.
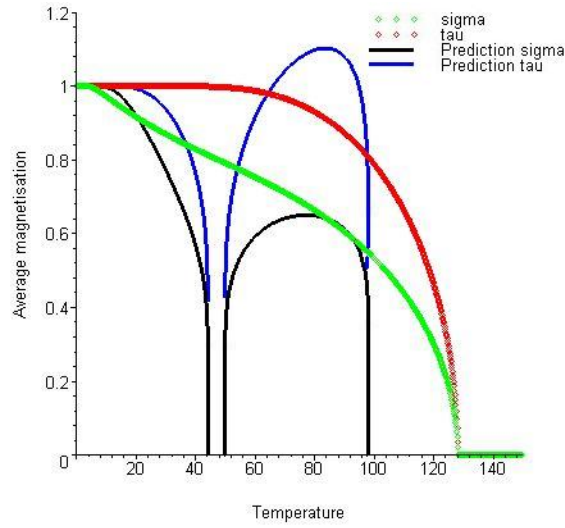
Figure 5.6: Graphs for the anisotropic ferromagnetic system on the Union Jack lattice with interactions $J = 100k_B$ and $K = -92k_B$. The data points are calculated using the coupled mean field equations and the solid lines are the Wu and Lin theorectical predictions. Note that the mean field curves do not see the second range of non-zero magnetisation. Equally the temperature is higher than all the critical temperatures.

This allows us to study systems with for example an external magnetic field. It does have its limitation however as it is not able to model metamagnetic systems due to the simplification used. Also we have seen that to model the Union Jack lattice models we require a coupled set of equations to produce results that more closely follow those of the known predictions. In addition we see that when the diagonal interactions are negative our mean field results do not closely follow the predictions. In the exact solution (4.8) the overall power is $1/8$, where in the mean field equations (5.11) the overall power is $1/2$. This limits the accuracy of the approximation. The approximation is an iterative process, and this could lead to any error being compounded. Thus a defect might lead to an instability, resulting in the accuracy of the approximation being poor.

Sampling Methods

As we have seen from the calculations of Section 3.2 and Chapter 4, it is not a practical possibility to calculate every state of the Ising model on a lattice of a reasonable size as it is NP complete. While one can use an approximation method like the mean field theory approach seen in the previous Chapter 5, these generally only have a qualitative use and we would like a method that could produce reasonably accurate qualitative results. With this in mind in this chapter we will briefly discuss the area of Markov Processes [GS92] and how these can be sampled using Monte Carlo Methods to produce our results in a reasonable timeframe. While this is going to be a short presentation, the reader is directed to books such as [LB05, BB95] for a more in depth development of this theory. Our development however will follow the flow of the development in Section 7 of [PB94]. In Chapter 8 we will use these methods to preform some numerical experiments on various Ising model systems.

6.1 Monte Carlo methods

In a Monte Carlo simulation [MU49] one does not attempt to simulate the dynamics of the system; instead the idea is to generate states $i, j, \ldots$ by a stochastic process such that the probability π_i of state i is that given by an approximate distribution (in our case the canonical distribution). In a production run of a simulation, N states are generated and the desired quantity x_i (energy, magnetisation, pressure, etc.) is calculated for each state. If the probabilities are correct, then

$$\langle x \rangle = \lim_{N \to \infty} \frac{1}{N} \sum_i x_i. \tag{6.1}$$

In our case we will calculate the canonical ensemble, so the probability will be given by

$$\pi_i = \frac{1}{Z}e^{-\beta E_i} \text{ where } Z = \sum_i e^{-\beta E_i}.$$

In the general case a Monte Carlo simulation consists of the following steps:

1. Choose an initial condition

2. Select a move

3. Accept or reject the move using a criterion based on a detailed balance

4. Repeat steps (2) and (3) until enough data is collected.

Typically the data from the early part of a run is discarded since the system will not have had enough time to reach equilibrium.

From this definition we now have two questions:

1. How does the computer generate the states?

2. How can we make sure the probabilities are correct?

6.2   Markov processes

Now let us consider our Ising model. If it has N positions then it will have 2^N microstates. Using the Hamiltonians given in (3.25), (3.59) and (4.1), it is easy to calculate the energy of one of these states. If we were to calculate the expected value with the canonical ensemble, we would use a random number generator to assign the values of spin to each position, weight the contribution of that microstate by $e^{-\beta E_i}$ and repeat until the expectation value had converged. This is inefficient since all the states would appear with equal probability, including those with such small weight that they do not contribute to the thermodynamical average.

Instead to make the process more efficient, we will be sampling the transition between the states. This is because it will be more likely to occur between those of more dominance to the system. The process we shall use to generate these states will be a *Markov Process.*

If the model starts in a given microstate i, it will move to state j with transition probability $P_{j \leftarrow i}$ that does not depend on the previous history of the model. If we assume the model to be under some fairly general conditions, processes after the passage of a transient would produce states with a unique steady-state probability distribution. This steady-state probability π_j is an eigenvector, with eigenvalue one, of the transition matrix:

$$\pi_j = \sum_i P_{j \leftarrow i} \pi_i. \tag{6.2}$$

The steady-state probabilities are unique if the matrix $P_{j \leftarrow i}$ is regular, which means that for some integer n all elements of $(P_{j \leftarrow i})^n$ are positive and non-zero. Physically, this restriction implies that it is always possible to go from one state to any other state in a finite number of steps. Exceptions are matrices that are block diagonal, for example

$$\begin{bmatrix} P_{1 \leftarrow 1} & P_{1 \leftarrow 2} & 0 & 0 \\ P_{2 \leftarrow 1} & P_{2 \leftarrow 2} & 0 & 0 \\ 0 & 0 & P_{3 \leftarrow 3} & P_{3 \leftarrow 4} \\ 0 & 0 & P_{4 \leftarrow 3} & P_{4 \leftarrow 4} \end{bmatrix}.$$

Since there is no way of going from states 1 or 2 to 3 or 4, the stationary probability distribution will depend on whether one started with one of the first two states or one of the last two.

6.3 The Metropolis-Hastings algorithm

Suppose we wish to determine the transition matrix $P_{i \leftarrow j}$ for our Ising model so that the steady-state distribution is

$$\pi(i) = \frac{\exp - \beta E_i}{Z} \tag{6.3}$$

where $\beta = 1/(k_B T)$ and Z is the partition function. A possible method of generating a sequence of states from an initial state is to pick a site α randomly and attempt to flip (change the sign of) its spin. The resulting state (which may be the same state i if the attempt to flip σ_α fails) we call j. Let $P_{j \leftarrow i}$ be the transition probability from i

to j. After n steps the transition probability $P_{f \leftarrow i}(n)$ is given by

$$P_{f \leftarrow i}(n) = \sum_{i_1, i_2, \ldots, i_{n-1}} P_{f \leftarrow i_{n-1}} P_{i_{n-1} \leftarrow i_{n-2}} \ldots P_{i_1 \leftarrow i}.$$

After many steps the system will approach a limiting distribution

$$\frac{\pi(m)}{\pi(j)} = \exp\left[-\beta \left\{E(m) - E(j)\right\}\right]$$

for all pairs of states m, j. We now also require the transition probabilities to be normalised

$$\sum_j P_{j \leftarrow m} = 1 \tag{6.4}$$

and to obey

$$\frac{P_{j \leftarrow m}}{P_{m \leftarrow j}} = \frac{\pi(j)}{\pi(m)} = \exp\left[-\left\{E(j) - E(m)\right\}\right]. \tag{6.5}$$

We find that

$$\pi(m) = \sum_j P_{m \leftarrow j} \pi(j). \tag{6.6}$$

This relation holds, by definition, for any Markov process. The first step above follows from the normalisation in (6.4), while the second step involves substituting (6.5). From (6.1) we see that (6.6) implies that $\pi(m)$ is a stationary probability distribution of the process. Equation (6.5) is called the *principle of detailed balance*. It can be shown that it is a sufficient condition for arriving at the correct limiting probability distribution, provided that the process for selecting moves does no contain any traps. That is, it should always be possible to get from any given microstate to any other microstate.

The simplest and most frequently used method of achieving a detailed balance is the *Metropolis algorithm* [MRR$^+$53]:

1. Pick a site α randomly

2. Compute the energy change $\Delta E = E(f) - E(i)$ that would occur if the spin at site α was flipped

3. If $\Delta E < 0$, flip the site at site α; if $\Delta E > 0$ flip this site with probability $\exp(-\beta \Delta E)$

4. Repeat steps (1) to (3) until enough data has been collected.

An alternative to (3) that is sometimes used is to flip the spin σ_α with probability $[\exp(\beta\Delta E) + 1]^{-1}$ regardless of the sign of ΔE. It is easy to see that (6.4) is satisfied in both cases for all possible pairs of states. The allowed states therefore will occur with the correct frequency if the simulation is run long enough to reach the steady state.

Calibration

In this chapter we will be calibrating the program written for the project (code given in Appendix B) by running simulations on a triangular lattice and comparing them to the known results given in [Ste64] and [Bax75]. This is an important step as it will allow us to see how accurately the program is able to simulate a system quantitatively, rather than just qualitatively as seen with mean field theory in Chapter 5.

7.1 Isotropic system

7.1.1 Ferromagnetic

In this first simulation we will investigate an isotropic system with all nearest neighbour interactions as one hundred times the Boltzmann constant and a three site interaction of zero. This is the simplest system on a triangular lattice, and contains a phase transition. This will give us a rough guide as to whether we can continue with more complicated models. The resultant graph is shown in Figure 7.1. We can see from this graph that our simulation shows a good correlation with the known results that we described earlier. Our simulation tracks the phase transition at under 400 Kelvin quite well. The correlation is good for both the magnetisation data and the three-site correlator data. There is little noise after the phase transition and the simulation results quickly move to a zero average magnetisation. While visible, the noise is of a magnitude that is much lower than the rest of the data.
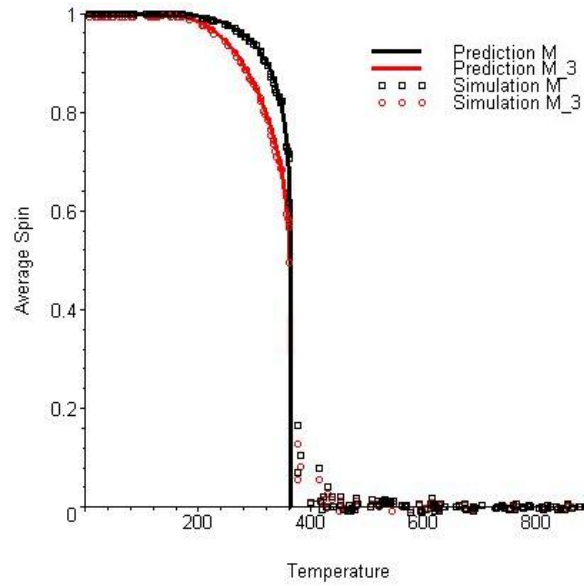
Figure 7.1: Graph of numerical simulation results plotted against the theoretical predictions of Stephenson [Ste64] and Baxter [Bax75] for a isotropic ferromagnetic system on the triangular lattice with interaction strengths $J_1 = 100k_B$, $J_2 = 100k_B$, $J = 100k_B$. The theoretical predictions are shown with solid lines. The numerical simulations are shown with points of the colour corresponding to the relative prediction.

7.1.2   Antiferromagnetic

Having looked at an isotropic ferromagnetic system, the next step is to look at an isotropic antiferromagnetic system. According to the conditions set out in the papers [Ste64, Bax75], we should expect results of zero average magnetisations. The graph we obtain from the simulations is shown in Figure 7.2. From the graph we see that the simulation data shows a good correlation with the predicted zero magnetisation, and appears to have little noise. We also note that there is no evidence of a phase transition across our temperature range. This is however a frustrated system. That is, not all of the interactions can be in their preferred states, one interaction must always be in a parallel state. To further investigate antiferromagnetic systems we have to move on to looking at anisotropic systems.
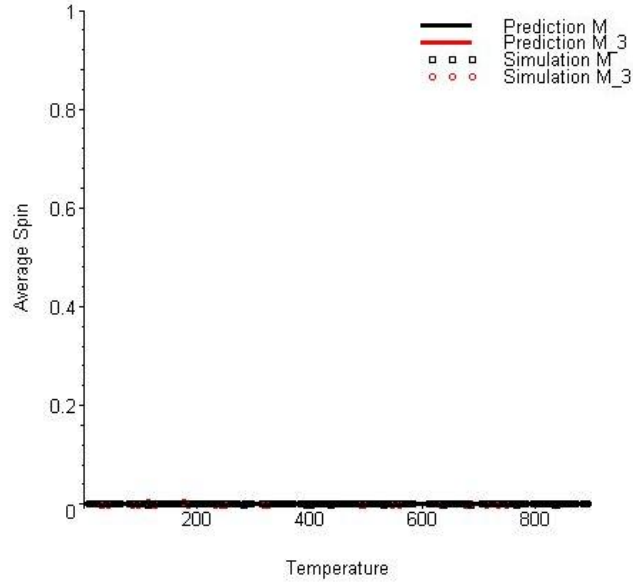
Figure 7.2: Graph of numerical simulation results plotted against the theoretical predictions of Stephenson and Baxter for the isotropic antiferromagnetic system on the triangular lattice with interaction strengths $J_1 = -100k_B$, $J_2 = -100k_B$, $J = -100k_B$.

7.2 Anisotropic systems

7.2.1 Ferromagnetic

Our investigation of anisotropic systems will start by trying to match the results given in Baxter's paper [Bax75] of the three-site interactions on the square lattice. We do this by setting the diagonal interaction strength to zero, while choosing values for the other two interactions. In our first simulation of these systems we will look at the ferromagnetic system where the horizontal and vertical interactions are one hundred times the Boltzmann constant. The graph we obtain is show in Figure 7.3: Once again our simulation has a good correlation with the know results. There is noise around the critical temperature, though the results quickly return to the expected value. The phase transition appears to be consistent with the known results.

7.2.2 Antiferromagnetic

Now we move to a more general anisotropic system in this set of systems, an antiferromagnetic system where the vertical interactions are of equal magnitude as the horizontal interactions but negative. Again we expect the results to show a zero aver-
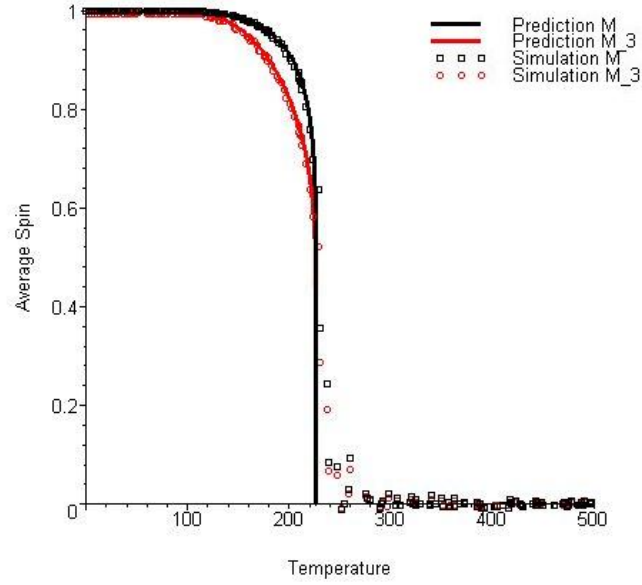
Figure 7.3: Graph of numerical simulation results plotted against Stephenson and Baxter theoretical predictions on the isotropic ferromagnetic system on the square lattice with interactions $J_1 = 100k_B$, $J_2 = 100k_B$, $J = 0$.

age magnetisation across the entire temperature range and no phase transitions. The graph of this simulation is show in Figure 7.4: As expected we can see that the simulation data is approximately zero average magnetisation and there is no sign of a phase transition in the temperature range. The noise seen is the data is low and seems constant across the data set. In this system we do not have the issue of frustrated bonds as the lack of the diagonal interactions allows all the interactions to be in their preferred state.

As an extension to the last simulation, we will now perform a simulation of a system of an antiferromagnetic system where the vertical and horizontal interactions are negative one hundred times the Boltzmann constant and the diagonal interactions are positive one hundred times the Boltzmann constant. The graph of this simulation is shown in Figure 7.5: Once again, as with the previous antiferromagnetic systems, we see that the data is around the zero average magnetisation. This shows a strong correlation to the known results for the triangular lattice, and the noise seems to be at a low level. There appears to be no phase transition across the data range, which is as expected. In this simulation the interactions will be in their preferred state. That is, along the horizontal and vertical bonds the spins will be in an anti-parallel arrangement, due to the ferromagnetic coupling of the diagonal bond.
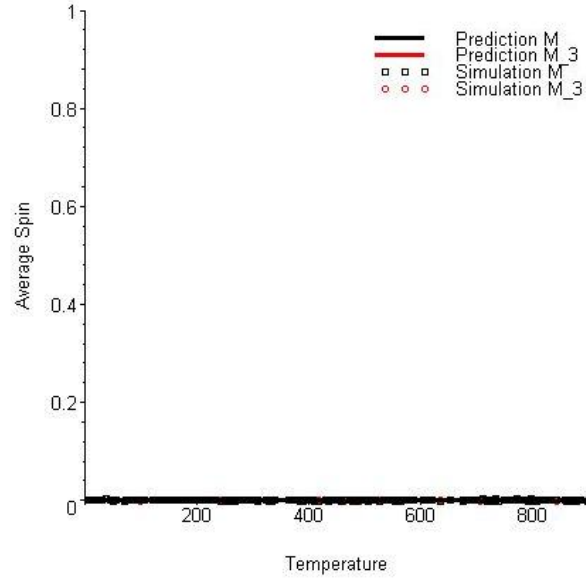
Figure 7.4: Graph of numerical simulation results plotted against the theoretical predictions from Stephenson and Baxter for the isotropic antiferromagnetic system on the square lattice with interactions $J_1 = 100k_B$, $J_2 = -100k_B$, $J = 0$.
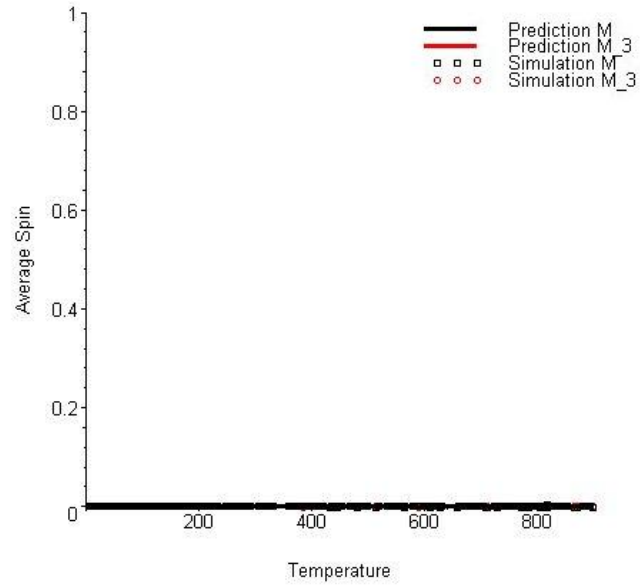
Figure 7.5: Graph of numerical simulation results plotted against the theoretical predictions from Stephenson and Baxter for the isotropic antiferromagnetic system on the square lattice with interactions $J_1 = -100k_B$, $J_2 = -100k_B$, $J = 100k_B$.

7.3   Overall result

In the simulations carried out during this chapter we have seen that the data produced shows a high correlation with the known results of Baxter and Stephenson. In producing accurate results for these systems, confidence can now be taken in future results as to their accuracy of modelling the system. In general we can say that the calibration of the program has gone well and a useful tool has been developed for the next chapter.

Numerical Simulations

Having calibrated our program against the triangular lattice, we move on to look at the Union Jack lattice. As we saw previously in Chapter 4, the theoretical predictions from Wu and Lin [WL89] sometimes give results that intuitively we do not expect. In this chapter we will perform numerical simulations on various systems and compare the results against Wu and Lin's predicted results, and the critical temperatures again against those from Vaks *et al.* [VLO66]. As in the triangular lattice simulations, we will be using a lattice of 100 sites by 100 sites in our program. The simulation results obtained will be for a finite lattice, while the theoretical predictions are for an infinite lattice. This lattice size is chosen as it is small enough to have a reasonable run time, while being large enough to suppress the finite size effects.

8.1 Isotropic ferromagnetic

Following the structure of our analysis in Section 4.3, we will start our numerical simulations with the isotropic ferromagnetic system. Here the interactions for our system will be $J = J' = J_n = 100k_B$. When we plot our simulation results against the predicted results of Wu and Lin we obtain Figure 8.1 below. As we can see, our simulation data has a high correlation with the prediction functions. As we have discussed before, Wu and Lin's result follows that of Vaks *et al.*, and so we can say that our simulation data also agrees with their phase predictions. There is some noise around the critical temperature, though is of a small magnitude when compared to the other results. After the noise part of the data, the points again follow the prediction with a high correlation.
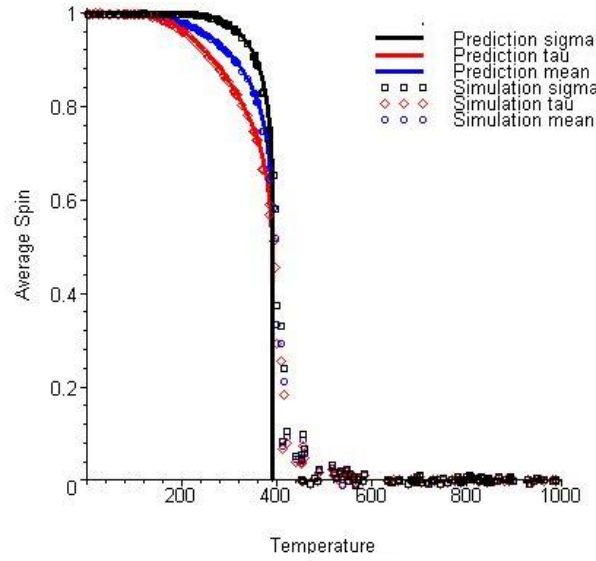
Figure 8.1: Graph of numerical simulation results plotted against theoretical predictions of Wu and Lin for an isotropic ferromagnetic system on the Union Jack lattice with interactions $J_1 = 100k_B$, $J_2 = 100k_B$, $J_3 = 100k_B$, $J_4 = 100k_B$, $J = 100k_B$, $J' = 100k_B$. Here the theoretical predictions are shown with the solid lines and the numerical results are shown with the points.

8.2   Anisotropic metamagnetic

We move on to the first system in which we saw a disagreement between the results of Wu and Lin and the configuration we expect. Our next simulation will be on an anisotropic metamagnetic system where $J_n = 10k_B$, $J = 100k_B$ and $J' = 100k_B$. The graph we obtain when we plot our simulation results against the predictions of Wu and Lin is shown in Figure 8.2 below. Our simulation results show a poor correlation to the prediction functions, with the average spin across the temperature being zero. From our analysis in Section 4.3 we concluded that the system should be antiferromagnetic when there is no external magnetic field. The results have a high correlation to an antiferromagnetic system, and so back up this analysis.

8.3   Anisotropic antiferromagnetic

Next we look at the anisotropic antiferromagnetic system. The system will have horizontal and vertical interactions of $J_n = 100k_B$ and diagonal interactions of $J = J' = -100k_B$ as before. The results are plotted against Wu and Lin's predictions in Figure 8.3 below. The correlation between the simulation data and Wu and Lin's prediction is
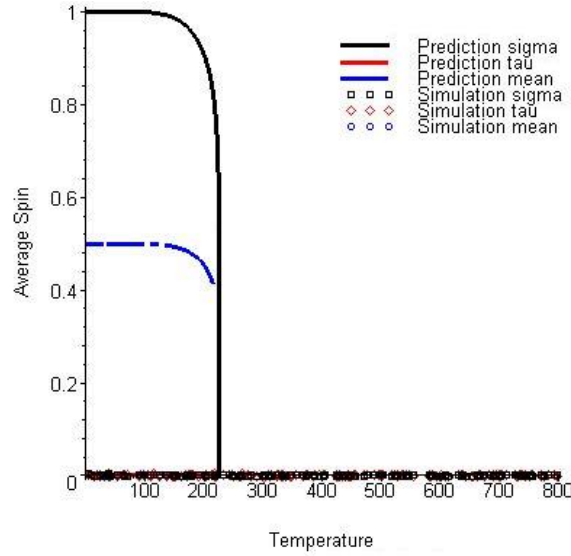
Figure 8.2: Graph of numerical simulation results plotted against theoretical predictions of Wu and Lin for the anisotropic metamagnetic system on the Union Jack lattice with interactions $J_n = 10k_B$, $J = 100k_B$ and $J' = -100k_B$. Note that the simulation results do not follow the curves of the prediction functions.
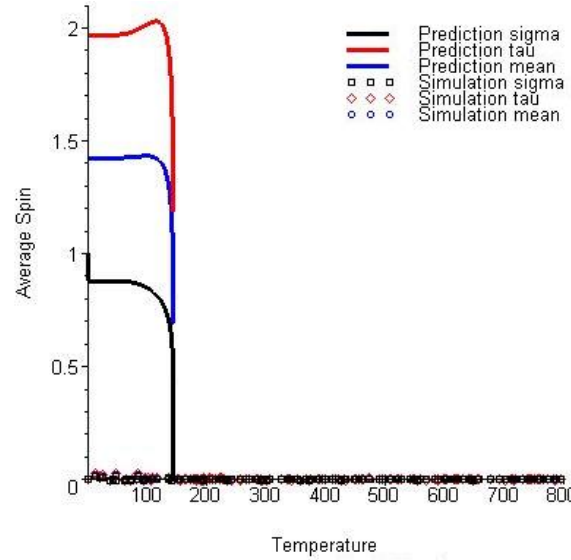
Figure 8.3: Graph of numerical simulation results plotted against theoretical predictions of Wu and Lin for an anisotropic antiferromagnetic system on the Union Jack lattice with interactions $J_1 = 100k_B$, $J_2 = 100k_B$, $J_3 = 100k_B$, $J_4 = 100k_B$, $J = -100k_B$, $J' = -100k_B$. Note that the simulation results do not follow the curves of the theoretical predictions and are physically possible.

again poor. Vaks *et al.* predict that this system should have average spin of zero. As we can see this confirms the results of the simulation. Overall the noise in the results is low and does not show any signs of detecting any non-zero magnetisation.

8.4   Anisotropic ferromagnetic

So far in simulations we have focussed on systems with only one dominant system. In Section 4.3 we saw that we can have an anisotropic ferromagnetic system where there is a re-entrant phase transition in the temperature range. The system we studied there was one with horizontal and vertical interactions of $J_n = 100k_B$ and diagonal interactions of $J = J' = -92k_B$. The graph of our simulation results plotted against the prediction functions of Wu and Lin is shown in Figure 8.4. We can see that our
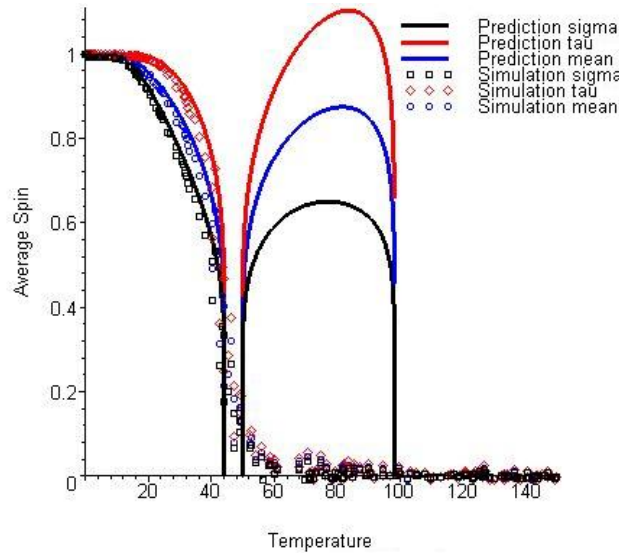
Figure 8.4: Graph of numerical simulation results plotted against theoretical predictions of Wu and Lin for an isotropic ferromagnetic system on the Union Jack lattice with interactions $J_1 = 100k_B$, $J_2 = 100k_B$, $J_3 = 100k_B$, $J_4 = 100k_B$, $J = -92k_B$, $J' = -92k_B$. Note that the simulation results have good correlation up to the first critical temperature, but do not show the re-entrant phase.

simulation results show good correlation with the predictions of Wu and Lin up to the first phase transition. Note that our simulation results do not show the re-entrant phase transition. As we discussed in Section 4.3, the re-entrant phase transition is between a disordered antiferromagnetic phase and an ordered antiferromagnetic phase. Due to the set up of the simulation program we are not able to see the difference

between the different antiferromagnetic phases. Compared to the prediction functions of Wu and Lin, we can see that our simulation results do not follow their predictions for the ordered antiferromagnetic phase, although do show an average magnetisation of zero.

Next in our study of the systems in Section 4.3, we will now go on to look at the anisotropic ferromagnetic system with horizontal and vertical interactions of $J_n = -100k_B$ and diagonal interactions of $J = J' = 92k_B$. The graph of this simulation is show in Figure 8.5. Here we see that our simulation results have good correlation with
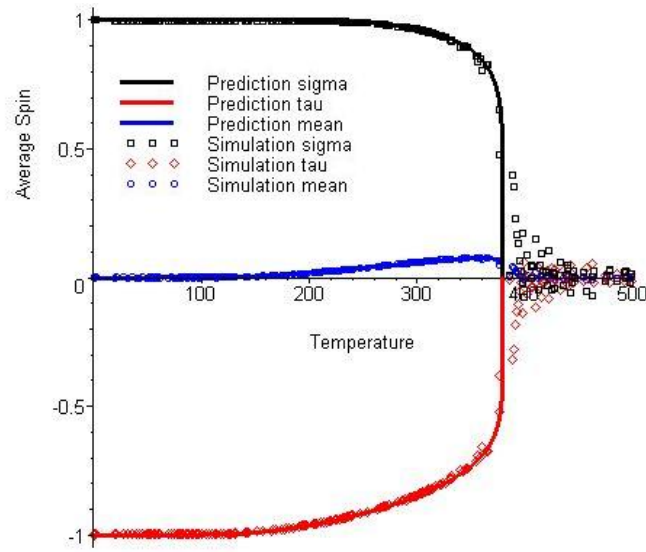
Figure 8.5: Graph of numerical simulation results plotted against theoretical predictions of Wu and Lin for an anisotropic ferromagnetic system on the Union Jack lattice with interactions $J_1 = -100k_B$, $J_2 = -100k_B$, $J_3 = -100k_B$, $J_4 = -100k_B$, $J = 92k_B$, $J' = 92k_B$. Note that the simulation results shows high correlation with both prediction curves.

the predicted functions of Wu and Lin. Our simulation also shows the slight overall magnetisation after 200 Kelvin, up to the critical temperature. There is noise after the critical temperature as the two sets of data converge to the average zero spin. We have seen this in all our ferromagnetic systems and it is not significant given the range of the rest of the data. So we have agreement with Wu and Lin's prediction.

Following on in our studies, we will look at the anisotropic systems where rotation produces different graphs. We first look at the system with horizontal interactions of $J_1 = J_3 = 100k_B/0.9^2$, vertical interactions of $J_2 = J_4 = 100k_B/0.9$ and diagonal interactions of $J = J' = 100k_B$, and the similar system rotated through 90 degrees.

The graphs of our simulation results are shown in Figure 8.6 below. As we see when
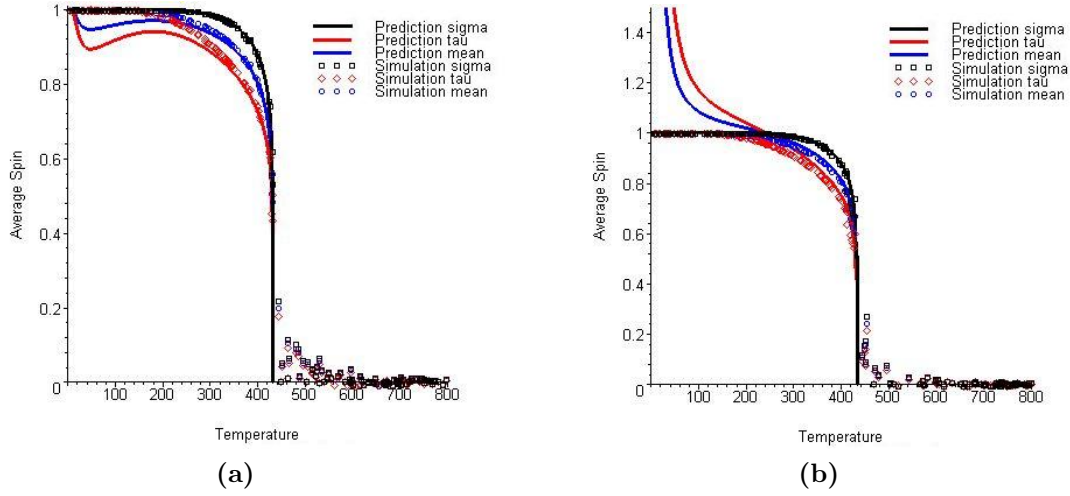
Figure 8.6: Graphs of numerical simulation results plotted against theoretical predictions of Wu and Lin for an anisotropic ferromagnetic system on the Union Jack lattice. (a) shows the system with interactions $J_1 = J_3 = 100k_B/0.9^2$, $J_2 = J_4 = 100k_B/0.9$ $J = J' = 100k_B$. (b) shows the rotated system with interactions $J_1 = J_3 = 100k_B/0.9$, $J_2 = J_4 = 100k_B/0.9^2$ $J = J' = 100k_B$. Note that although the prediction functions are different, the simulation results are identical. Also there is poor correlation with the τ-sublattice prediction.

we compare the simulations results, the data forms similar curves and has a similar phase transition at equal critical temperatures. In comparison to the predictions of Wu and Lin, we see that at the higher temperatures the data follows all three curves with good correlation. At lower temperatures, below about 200 Kelvin, we see that the σ prediction still has good correlation for both systems. This confirms that the prediction for τ (and subsequently the mean magnetisation) is not correct below this temperature. It also shows that rotating the lattice should not have an effect on the results of the system.

A further result can be seen if we now take a system similar to the previous example but with negative diagonal interactions. For an example of this type of system we will look at the system with horizontal interactions of $J_1 = J_3 = 100k_B/0.9^2$, vertical interactions of $J_2 = J_4 = 100k_B/0.9$ and diagonal interactions of $J = J' = -100k_B$, and the same system rotated through 90 degrees. The graphs of our simulation results are shown in Figure 8.7. Again we see as in Figure 8.6 that the simulation results are very similar to each other. However, in this case we see that all three simulation results have a range of lower results at low temperatures. After these lower results the
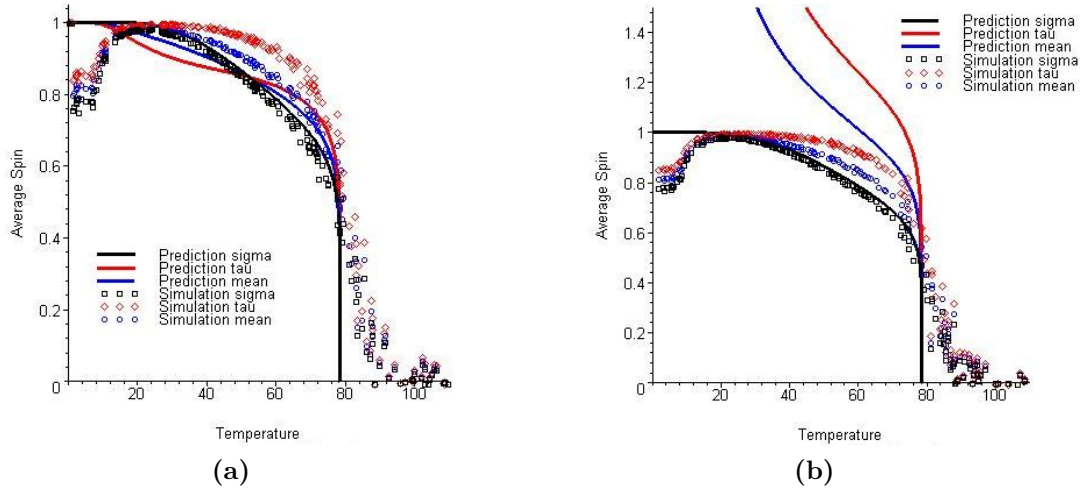
Figure 8.7: Graphs of numerical simulation results plotted against theoretical predictions of Wu and Lin for an anisotropic ferromagnetic system on the Union Jack lattice. (a) shows the system with interactions $J_1 = J_3 = 100k_B/0.9^2$, $J_2 = J_4 = 100k_B/0.9$ $J = J' = -100k_B$. (b) shows the rotated system with interactions $J_1 = J_3 = 100k_B/0.9$, $J_2 = J_4 = 100k_B/0.9^2$ $J = J' = -100k_B$. Note that although the prediction functions are different, the simulation results are identical. Also there is slight deviation at low temperatures of the simulation results.

simulations then move up again to the prediction curves, following the σ prediction in both cases. We do see that in both cases the simulation results and predicted results show the same critical temperature and phase transition. This oddity maybe due to the simulation being performed on a finite system, while the theoretical predictions are for an infinite lattice.

8.5 Magnetic fields

One advantage of using a numerical simulation, is that we can study systems in external magnetic fields. Metamagnetic systems have the property that when we add a external magnetic field we obtain results that show average magnetisation of a higher magnitude than that of the applied field. In all the following systems, the base system is one where the square lattice interactions are $J_n = 10k_B$ and the diagonal interactions are $J = -J' = 100k_B$, our standard metamagnetic system. In our first simulation we will apply a magnetic field of strength equal to the Boltzmann constant. The graph is show in Figure 8.8. With this magnetic field we can see already that at low temperatures that there is now a non-zero average magnetic spin. Also the sublattices now have distinct
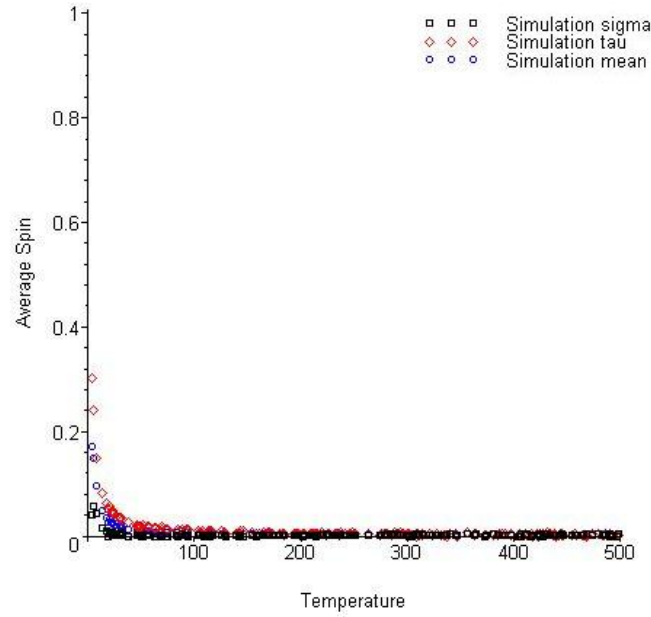
Figure 8.8: Graph of the simulation results for the anisotropic metamagnetic system on the Union Jack lattice in an external magnetic field of strength k_B. Note that at low temperatures there is non-zero magnetisation of greater magnitude than the external magnetic field.

curves at these temperatures, rather than being in the disorder antiferromagnetic state. The average magnetisation is small in comparison with the possible values for the spin states. So our next step is to increase the magnetic field to a strength ten times the Boltzmann constant. The graph we obtain is shown in Figure 8.9 below. We can see again that a non-zero average magnetisation has resulted. We see also that the σ prediction is now showing more of a sign of a ferromagnetic curve, while the τ prediction is a less sharp curve. Above 200 Kelvin we see a slight increase in the magnetisation suggesting that this is the critical temperature where the system is going into a disordered phase. When we look back to Figure 4.5 we see that this is also the point where Wu and Lin's prediction hits the x-axis.

For completeness we will now look at the same metamagnetic system in an external magnetic field of $100k_B$. The graph of the results we obtain is shown in Figure 8.10. Here we see that the results are showing more of a ferromagnetic shape. Again above 200 Kelvin the sublattice results are converging to a central value, and all the curves seem to be related to each other. Below this temperature we see that the τ-sublattice is showing more of a ferromagnetic phase, although the curve is not sharp. Meanwhile the σ-sublattice has a shape suggesting a ferromagnetic phase at low temperatures but
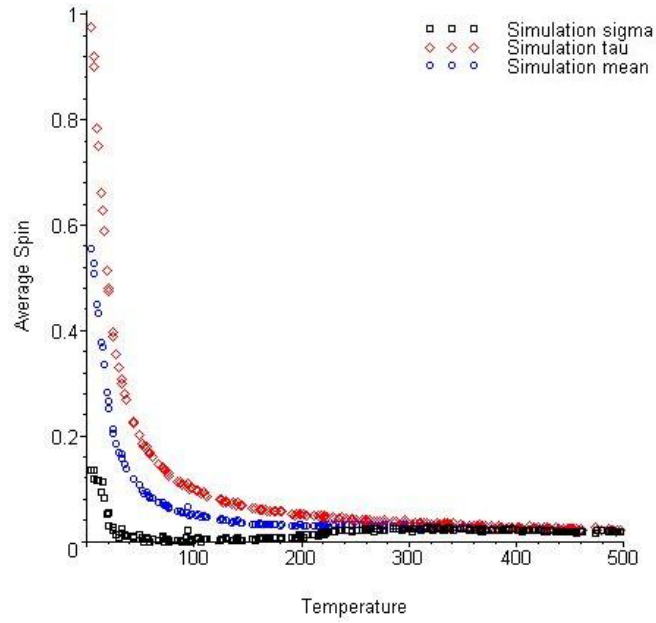
Figure 8.9: Graph of the simulation results for the anisotropic metamagnetic system on the Union Jack lattice in an external magnetic field of strength $10k_B$.
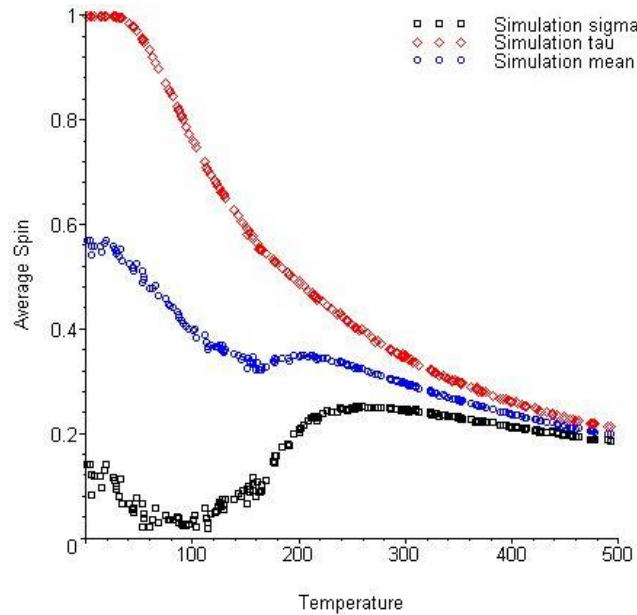
Figure 8.10: Graph of the simulation results for the anisotropic metamagnetic system on the Union Jack lattice in an external magnetic field of strength $100k_B$.

moves up to a maximum at 200 Kelvin.

8.6   Summary

We have seen in this chapter that our simulation results help support our findings in Chapter 4. We have good correlation between our simulation results in isotropic ferromagnetic and most anisotropic ferromagnetic systems and Wu and Lin's predictions [WL87, WL89] while in a ferromagnetic phase. However in terms of the τ-sublattice prediction we saw that our simulations support that the results should not change by a 90 degree lattice rotation. We also saw that our simulation agrees with Valks *et al.* predictions [VLO66] for the antiferromagnetic systems. In addiction we showed that the re-entrant phase transition can not be seen from average magnetisation alone.

In addition we showed that our simulation can be used to further investigate system properties that occur in external magnetic fields. This shows that we have a powerful tool for numerical simulation of the Union Jack lattice.

Conclusion

Over the course of the thesis we have seen two main results. The re-entrant phase transition can not be seen when looking at the average magnetisation results. In addition we have seen that the prediction functions of Wu and Lin in their papers [WL87, WL89] produce some anomalous results. In this section we will review the results from the different approaches we undertook and discus the disagreements from Wu and Lin's results that were found. At the end of the section we will discuss the additional conditions that when applied allow the prediction functions of Wu and Lin to show a greater agreement with known theories.

Our aim was to investigate the properties of the model and specifically the re-entrant phase transitions. From our investigations we have seen that these re-entrant phase transitions can not be seen when only looking at the average magnetisation results. This is due to the transition being from an unordered antiferromagnetic phase to an ordered antiferromagnetic phase. While this can been seen in the Vaks *et al.* paper [VLO66], this can be seen from other order parameters, the average magnetisation of an unordered antiferromagnetic phase and an ordered antiferromagnetic phase is identically zero. This was further confirmed by our numerical simulations, which also did not see these re-entrant phase transitions due to being focused on the average magnetisation of the system. However we have seen that the predictions of Wu and Lin can be used to find the critical temperatures of these phase transitions, and agree with the calculations of Vaks *et al.*

In addition, we have seen that Wu and Lin's prediction for the σ-sublattice given in [WL87] requires additional conditions to agree with our numerical simulations. It is possible to classify the phases of the system by examining the γ terms of equation (4.9). Wu and Lin's predictions under the current conditions produce non-zero magnetisations

for non-ferromagnetic systems. However if we impose the conditions that we only use the prediction formula (4.8) when $\gamma_1 < 0$ or $\gamma_1 \ \gamma_2 \ \gamma_3 \ \gamma_4 \ > 0$ and it is zero outside those conditions, their results now work for all systems.

When we look at Wu and Lin's prediction for the τ-sublattice, given in [WL89], we see for the general anisotropic lattice there are some additional issues. When the interactions on the square lattice are equal, by applying the conditions above we can eliminate the physically impossible results that we saw in the antiferromagnetic and metamagnetic phases. This is in agreement with the earlier work of Lin and Wang [Lin88]. When we consider unequal vertical and horizontal interactions on the square lattice we see that physically plausible results are only possible when the horizontal interactions are larger than the vertical interactions. From Figure 8.6 we see however that our simulation results show that there is still some disagreement at low temperatures. While this suggests that further conditions are required for this prediction, they would be more involved than those for the σ-sublattice. For the current prediction, imposing the condition that $J_1 = J_2 = J_3 = J_4$, together with the conditions imposed on σ would bring agreement with the simulation results.

From our investigation of mean field theory, we saw that it is a poor quantitative predictor for the systems. In addition it has the limitation of not being able to look at metamagnetic systems. Qualitatively however, it does allows a prediction of the general behaviour of symmetric systems without a re-entrant phase transition.Again the mean field prediction does not see re-entrant phase transitions in terms of the magnetisation. In these systems its prediction of critical temperature is generally in line with the third phase transition critical temperature. The results were useful for confirming those the results we found in our theoretical analysis.

In conclusion, in the thesis we have not been able to investigate the re-entrant phase transition using the average magnetisation. However, we have developed an accurate simulation program for finite systems for both the triangular and Union Jack lattices. This simulation program has allowed us to investigate inconsistencies in the theoretical predictions of Wu and Lin [WL87, WL89]. We have also shown that it is possible with additional conditions to make these predictions model the systems more accurately.

Future research could extend this work to look more closely at the rotational variance that is observed in the τ-sublattice prediction. Intuitively the predictions should be invariant under rotation of the lattice, as shown in the simulation results, and this

disagreement requires further investigation. Equally as we saw in Figure 8.7 that there is an oddity in the results at low temperatures, and this should be investigated to determine if it is affected by the lattice size.

As well as looking to the current lattice configuration, further research could take place to see if the results found occur in other configurations. Research into the hexagonal and double hexagonal lattices may be interesting to determine if a re-entrant phase is present. In addition in terms of the Union Jack lattice, an investigation into mixed spin lattices such as those of Strečka *et al.* [Str06, SvD06] may bring insight into both the re-entrant phase transitions and the metamagnetic systems.

In terms of the simulation program, other Monte Carlo methods could be investigated to improve its efficiently. After a conversation with Dr Tim Garoni of the University of Melbourne at the Australian Mathematics Society meeting in 2010, the algorithm of Swendsen and Kotecky [SW87], which has been improved by Wang, Swendsen and Kotecky in [WSK90], was suggested to be a more efficient method. As this is a cluster method, its use on a Union Jack lattice would need to be investigated. If the algorithm could be used on this lattice type, it would overcome the problem of non-ergodic data in low temperature antiferromagnetic systems. It would also allow the simulation to not be affected by the critical slowing down effects around the second-order phase transitions.

References

[Bar82] F. Barahona, *On the Computational Complexity of Ising Spin Glass Models*, Journal of Physics A: Mathematical and General **15** (1982), 3241–3253.

[Bax75] R J Baxter, *Triplet order parameter of the triangular Ising model*, Journal of Physics A: Mathematical and General **8** (1975), no. 11, 1797–1805.

[Bax86] ———, *Free-Fermion, Checkerboard and Z-Invariant Lattice Models in Statistical Mechanics*, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences **404** (1986), 1–33.

[Bax89] ———, *Exactly solved models in statistical mechanics*, Academic Press, 1989.

[BB95] K Binder and A Baumgrtner, *The Monte Carlo Method in Condensed Matter Physics*, 2 ed., Springer, 1995.

[BC91] C.F. Baillie and P.D. Coddington, *Comparison of cluster algorithms for two-dimensional Potts models*, Physical Review B **43** (1991), 10617–10621.

[BD06] D Bray and T Duke, *Conformational spread: The propagation of allosteric states in large multiprotein complexes*, Rendiconti Lincei **17** (2006), 221–241.

[Bet35] H. A. Bethe, *Statistical Theory of Superlattices*, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences **150** (1935), 552–575.

[BHL02] H.W.J. Blöte, J.R. Heringa, and E. Luijten, *Cluster Monte Carlo: Extending the range*, Computer Physics Communications **147** (2002), 5863.

[Bru67] S G Brush, *History of the Lenz-Ising Model*, Reviews of Modern Physics **39** (1967), no. 1, 883–893.

[BW34] W. L. Bragg and E. J. Williams, *The Effect of Thermal Agitation on Atomic Arrangement in Alloys*, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences **145** (1934), 699–730.

[BW73] R J Baxter and F. Y. Wu, *Exact Solution of an Ising Model with Three-Spin interactions on a Triangular Lattice*, Physical Review Letters **31** (1973), no. 21, 1294–1297.

[Ear68] A Earnshaw, *Introduction to magnetochemistry*, Academic Press, 1968.

[Fey72] R Feynman, *Lectures on Statistical Mechanics*, Addison-Wesley, 1972.

[FG40] R. H. Fowler and E. A. Guggenheim, *Statistical Thermodynamics of Super-Lattices*, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences **174** (1940), 189–206.

[FW70] C Fan and F. Y. Wu, *General Lattice Model of Phase Transitions*, Physical Review B **2** (1970), no. 3, 723–733.

[Gre62] H. S. Green, *The Long-Range Correlations of Various Ising Lattices*, Zeitschrift für Physik **171** (1962), t29–148.

[GS92] G Grimmett and D Stirzaker, *Probability and Random Processes*, Oxford University Press, 1992.

[Hei28] W. Heisenberg, *Zur Theorie des Ferromagnetismus*, Zeitschrift für Physik A Hadrons and Nuclei **49** (1928), 619–636.

[Isi25] E Ising, *Beitrag zur Theorie des Ferromagnetismus*, Zeitschrift für Physik **31** (1925), 253–258.

[Ist00] S Istrail, *Statistical Mechanics, Three-Dimensionality and NP-completeness - I. Universality of Intractability for the Partition Function of the Ising Model Across Non-Planar Lattices*, Proceedings of the thirty-second annual ACM symposium on Theory of computing, 2000, pp. 87–96.

[Kau49] B Kaufman, *Crystal Statistics. II. Partition Function Evaluated by Spinor Analysis*, Physical Review **76** (1949), 1232 – 1243.

[KIUH65] R Kubo, H Ichimura, T Usul, and N Hashizume, *Statistical Mechanics, An Advanced Course*, North Holland, 1965.

[KW41a] H. A. Kramers and G. H. Wannier, *Statistics of the Two-Dimensional Ferromagnet. Part I*, Physical Review. **60** (1941), 252–262.

[KW41b] ———, *Statistics of the Two-Dimensional Ferromagnet. Part II*, Physical Review **60** (1941), 263–276.

[KW52] M. Kac and J. C. Ward, *A Combinatorial Solution of the Two-Dimensional Ising Model*, Physical Review **88** (1952), 1332 – 1337.

[LB05] D P. Landau and K Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*, second ed., Cambridge University Press, 2005.

[LCLG94] D. L. Lin, Hao Che, Wei Lai, and Thomas F. George, *Critical temperature of Ising films with cubic lattices*, Physical Review E **49** (1994), 2155–2160.

[Lin88] K. Y. Lin, *Spontaneous Magnetization Of The Ising Model on a 4-8 Lattice*, Physics Letters A **128** (1988), 35–38.

[LL80] L. D. Landau and E. M. Lifshitz, *Statistical Physics Parts 1 and 2*, Pergamon Press, 1980.

[LS74] L L. Liu and H. E Stanley, *Exact Solution in an External Magnetic Field of Ising Models with three-spin interactions*, Physical Review B **10** (1974), no. 7, 2958–2961.

[LW87] K.Y. Lin and S.C. Wang, *Spontaneous Magnetization Of The Ising Model On A Union Jack Lattice*, Physics Letters A **125** (1987), no. 1, 3–4.

[LW88] ———, *Spontaneous Magnetization Of The Ising Model On The Union Jack and 4-6 Lattices*, Physics Letters A **128** (1988), 143–145.

[Man88] F. Mandl, *Statistical Physics*, second edition ed., The Manchester Physics Series, John Wiley & Sons Ltd., 1988.

[MRR+53] N Metropolis, A W. Rosenbluth, M N. Rosenbluth, A H. Teller, and E Teller, *Equation of State Calculations by Fast Computing Machines*, The Journal of Chemical Physics **21** (1953), no. 6, 1087–1092.

[MU49] N. Metropolis and S Ulam, *The Monte Carlo Method*, Journal of the American Statistical Association **44** (1949), no. 247, 335–341.

[NM53] G F. Newell and E W. Montroll, *On the Theory of the Ising Model of Ferromagnetism*, Reviews of Modern Physics **25** (1953), 353 – 389.

[Ons44] L. Onsager, *Crystal Statistics. I. A Two-dimensional model with an Order-Disorder Transition*, Physical Review **65** (1944), 117–149.

[Orc03] A. F. Orchard, *Magnetochemistry*, Oxford University Press, 2003.

[PB94] M Plischke and B Bergersen, *Equilibrium Statistical Physics*, second edition ed., World Scientific, 1994.

[Pei36] R. E. Peierls, *On Ising's Model of Ferromagnetism*, Mathematical Proceedings of the Cambridge Philiosophical Society, vol. 32, 1936, pp. 477–481.

[Pin68] D A Pink, *Three-site correlation functions of the two-dimensional Ising model*, Canadian Journal of Physics **46** (1968), 2399–2405.

[Pot52] R. B. Potts, *Spontaneous Magnetization of a Triangular Ising Lattice*, Physical Review **88** (1952), 352.

[SAL90] D F. Shriver, P.W. Atkins, and C H. Langford, *Inorganic Chemistry*, Oxford University Press, 1990.

[Sha92] A. G. Sharpe, *Inorganic Chemistry*, 3rd edition ed., Longman, 1992.

[SML64] T. D. Schultz, D. C. Mattis, and E. H. Lieb, *Two-Dimensional Ising Model as a Soluble Problem of Many Fermions*, Reviews of Modern Physics **36** (1964), no. 3, 856–871.

[SSSEA06] M. A. Sumour, D. Stauffer, M. M. Shabat, and A. H. El-Astal, *Test of Universality in Anisotropic 3D Ising Model*, Physica A: Statistical Mechanics and its Applications **368** (2006), no. 1, 96–100.

[Ste64] J Stephenson, *Ising Model Spin Correlations on the Triangular Lattice*, Journal of Mathematical Physics **5** (1964), no. 8, 1009–1024.

[Ste70] _____, *Ising Model with Antiferromagnetic Next-Nearest-Neighbour Coupling: Spin Correlations and Disorder Points*, Physical Review B **1** (1970), no. 11, 4405–4409.

[Str06] J Strečka, *Weak universality, bicritical points and reentrant transitions in the critical behaviour of a mixed spin-1/2 and spin-3/2 Ising model on the Union Jack (centered square) lattice*, Physica Status Solidi (b) **243** (2006), no. 3, 708715.

[Suz76] M Suzuki, *Relationship between d-Dimensional Quantal Spin Systems and (d + 1)-Dimensional Ising Systems*, Progress of Theoretical Physics **56** (1976), no. 5, 1454–1469.

[Suz85] _____, *Transfer-matrix method and Monte Carlo simulation in quantum spin systems*, Physical Review B **31** (1985), no. 5, 2957–2965.

[SvD06] J Strečka, L Čanová, and J Dely, *Weak universal critical behaviour of the mixed spin-(1/2, s) Ising model on the Union Jack (centered square) lattice: integer versus half-odd-integer spin-S case*, Physica Status Solidi (b) **243** (2006), no. 8, 19461955.

[SW75] J E Sacco and F Y Wu, *32-vertex model on the triangular lattice*, Journal of Physics A: Mathematical and General **8** (1975), 1780–1788.

[SW87] R.H. Swendsen and J.-S. Wang, *Nonuniversal critical dynamics in Monte Carlo simulations*, Physical Review Letters **58** (1987), 86–88.

[tH66] D. ter Haar, *Elements of Thermostatistics*, Holt, Rinehart and Winston, New York, 1966.

[VLO66] V. G. Vaks, A. I. Larkin, and Yu. N. Ovchinnikov, *Ising Model With Interaction Between Nonnearest Neighbors*, Soviet Physics JETP **22** (1966), no. 4, 820–826.

[VS95] Angelica M. Vecchio-Sadus, *Electrochemical synthesis and magnetic studies of neutral transition metal imidazolates and pyrazolates*, Transition Metal Chemistry **20** (1995), 46–55.

[Wil61] J Wilks, *The Third Law of Thermodynamics*, Oxford University Press, 1961.

[WL87] F. Y. Wu and K. Y. Lin, *Ising Model on the Union Jack Lattice as a Free Fermion Model*, Journal of Physics A: Mathematical and General **20** (1987), 5737–5740, Corrigendum: [WL88].

[WL88] ———, *Corrigendum: Ising Model On The Union Jack Lattice As A Free Fermion Model, Wu F Y And Lin K Y 1987 J. Phys. A: Math. Gen.* **20** **5737-40**, Journal of Physics A: Mathematical and General **21** (1988), 1739.

[WL89] ———, *Three Spin Correlation of the Free-Fermion Model*, Journal of Physics A: Mathematical and General **22** (1989), 1121–1130.

[Wol89] U. Wolff, *Collective Monte Carlo Updating for Spin Systems*, Physical Review Letters **62** (1989), 361–364.

[WSK90] Jian-Sheng Wang, Robert H. Swendsen, and Roman Kotecky, *Three-state antiferromagnetic Potts models: A Monte Carlo study*, Physical Review B **42** (1990), no. 4, 2465–2474.

[Yan52] C N Yang, *The Spontaneous Magnetization of a Two-Dimensional Ising Model*, Physical Review **85** (1952), 808–816.

[Yew70] R H. T. Yew, *Composite Ising Lattices. I. Bragg-Williams Approximation*, Physical Review B **1** (1970), no. 3, 1180–1183.

Constants and Conversion Factors

Boltzmann Constant (k_B) ....$8.617343 \times 10^{-5}\text{eVK}^{-1}$

Avogadro's number ....$6.02 \times 10^{23}\text{mol}^{-1}$

1 electron-volt (eV) ....$1.60 \times 10^{-19}\text{J}$

or $1.1605 \times 10^{4}\text{K}$

0 Kelvin ....$-273.16°\text{C}$

Programming I

B.1 Programmer's guide

The broad purpose of this code is to simulate the Ising Model on a two-dimensional lattice with specific configurations. It does this by first asking the user for the initial conditions of the system along with the configuration of the lattice, the temperature range required and how many initial points to ignore. The program then simulates the model over randomly chosen temperature points, and outputs the data to a text file. The program has the ability to be split the output into sublattices if required.

B.1.1 Flow

Initially the user is asked for the type of lattice, the interaction strengths, external magnetic field strength they want the system to be running with, the number of points to ignore and the temperature range they are interested in. After all the conditions have been entered the system is initialised with all the sites being set to the "UP" (+1) position. The initial energy is then calculated for the system. Then a temperature is picked at random within the range given by the user, and the system is simulated at this temperature. Initially the first set of data points are ignored up to the level required by the user. Then the next points are averaged and stored as the average spin value for that temperature. The user is then told the overall average spin. The system is reset to the "UP" position for every site and is simulated again at a random temperature, not equal to the past temperatures. After 200 data points have been stored in the array, it is then outputted in a suitable format to a text file. The program then ends, telling the user how long the simulation took.
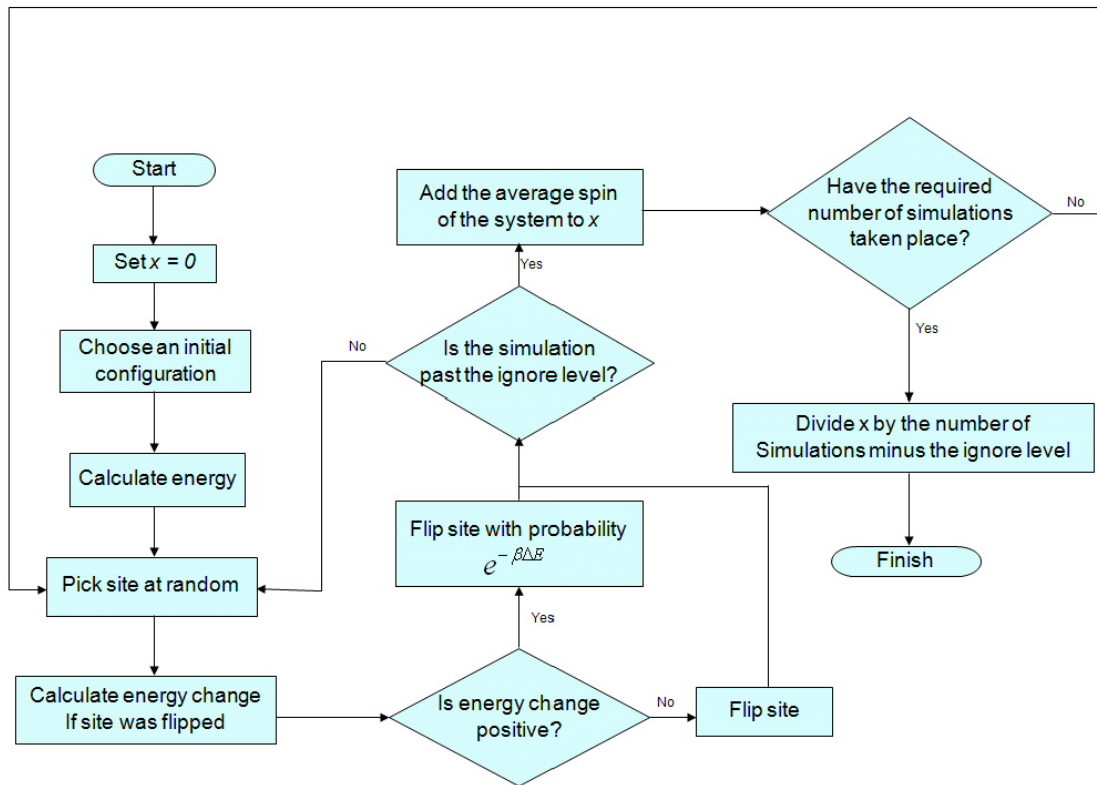
Figure B.1: Flowchart of the numerical simulation program

B.2 Code listings for main program

B.2.1 isling2dt.h

```
1   %auto−ignore
2   // standard C headers
3   #include <stdio.h>
4   #include <math.h>
5   #include <stdlib.h>
6   #include <time.h>
7
8   // Some useful system variables
9   enum {
10      UP= 1,
11      DOWN= −1,
12      ITERATIONS= 10000,
13      MODELS= 1000,
14      COND= 1000,
15      length= 100,
16      height= 100,
17      array_len= 200
18  };
```

```
19
20   #define BOLT 8.617343*pow(10,-5)     /*The boltzman contant*/
21
22   typedef struct actions {
23           double jh;
24           double jh2;
25           double jv;
26           double jv2;
27           double jdu;
28           double jdd;
29           double h;
30           double ku;
31           double kd;
32           char type;
33   } ACT;
34
35   typedef struct coordinate {
36           float temp;
37           double p;
38           double q;
39           double r;
40           double tripu;
41           double tripd;
42           struct coordinate *next;
43   } COORD;
44
45   //Prototypes of our functions
46   int setup_chain(char *);
47   double setup_gen(char *, char *);
48   void initialise_chain(int [length][height]);
49   void *safe_malloc(size_t, char *);
50   void *safe_realloc (int *, size_t, char *); /* Does the same for realloc */
51   void metropolis(int [length][height], float, ACT, double [3], int [3]);
52   double energy(int [length][height], ACT);
53   double energyb(int [length][height], ACT, int, int, int);
54   void write_2d_array (char *, ACT);
55   void set_temp(float , float);
56   void free_cond (void);              /* Frees the memory taken by linked list */
57
58   extern COORD *hol1;
```

B.2.2   isling2dt.c

```
1   %auto−ignore
2   /* We link our files with a standard header */
3   #include "isling2dt.h"
4
5   COORD *hol1= NULL;
6
7   int main() {
8           // intialising variables
9           int **chain; // for our lattice
10          int i, point, count, cycle, ignore, hours, minutes, seconds, *lattice;
11          int timeleft;
12          time_t timer;
13          char input, save;
14          COORD *coord;
15          double j, k;
16          float min, max, minmax, percent;
17          ACT vars;
18          double *split;
19          char filename[100];
20          min= 0;
21          max= min;
22          // some intial values incase of square lattices
23          vars.ku= 0;
24          vars.kd= 0;
25          vars.jv= 0;
26          vars.jh= 0;
27          vars.jdu= 0;
28          vars.jdd= 0;
29          // Finding out the type of lattice required
30          while(vars.type != 'S' && vars.type != 'T' && vars.type != 'U' && vars.type !=
                'N' && vars.type != 'H' && vars.type != 'D') {
31                  printf("What type of lattice would you like to study? \n ([S]quare, [T
                        ]riangular, [U]nion Jack, [N]ext Nearest, [H]exagonal, [D]ouble
                        Hexagonal) ");
32                  fscanf(stdin, "%1s", &vars.type);
33                  vars.type= toupper(vars.type);
34          }
35          while(input != 'A' && input != 'I') {
36                  printf("Is the lattice [I]somorpic of [A]ntimorphic? (I or A) ");
37                  fscanf(stdin, "%1s", &input);
38                  input= toupper(input);
39          }
40          // Finding out the attraction forces between particles
41          if(input == 'I') {
42                  // for the isotopic lattice
43                  j= setup_gen("the attraction \n between two particle", "real");
44                  vars.jh= j;  // if isotopic all j values are the same
45                  vars.jv= j;
```

```
46                    vars.jh2= j;
47                    vars.jv2= j;
48                    if(vars.type != 'S' && vars.type != 'H' && vars.type != 'D') {
49                            // specfic qualities that are different from square lattice
50                            vars.jdu= j;
51                            k= setup_gen("the attraction between three particles", "real")
                                    ;
52                            vars.ku= k;      // as all values of k are the same too
53                            if(vars.type == 'U' || vars.type == 'N') {
54                                    // specfic qualities for Union Jack and NN lattices
55                                    vars.jdd= j;
56                                    vars.kd= k;
57                            }
58                    } else if(vars.type == 'D') {
59                            k= setup_gen("the attraction between two particles second
                                    hexagon", "real");
60                            vars.ku= k;
61                            vars.kd= k;
62                    }
63            } else if(input == 'A') {
64                    // for the antitopic lattice
65                    vars.jh= setup_gen("the attraction \n between two particle, horizontal
                            ", "real");
66                    vars.jv= setup_gen("the attraction \n between two particle, vertical",
                            "real");
67                    vars.jh2= vars.jh;
68                    vars.jv2= vars.jv;
69                    if(vars.type != 'S' && vars.type != 'H' && vars.type != 'D') {
70                            // specfic qualties that are different from square lattices
71                            vars.jdu= setup_gen("the attraction \n between two particle,
                                    diagonal up", "real");
72                            vars.ku= setup_gen("the attraction \n between three particles
                                    up", "real");
73                            if(vars.type == 'U' || vars.type == 'N') {
74                                    // specfic qualities for the Union Jack lattice
75                                    if(vars.type == 'U') {
76                                            printf("Does J1 = J3 AND J2 = J4?: ");
77                                            fscanf(stdin,"%1s",&save);
78                                            printf("\n");
79                                            save= toupper(save);
80                                            if(save == 'N') {
81                                                    vars.jh2= setup_gen("the attraction \n
                                                            between two particle, horizontal
                                                            (J3)", "real");
82                                                    vars.jv2= setup_gen("the attraction \n
                                                            between two particle, vertical (
                                                            J4)", "real");
83                                            }
84                                    }
```

```
85                                     vars.jdd= setup_gen("the attraction \n between two
                                              particle, diagonal down", "real");
86                                     vars.kd= setup_gen("the attraction \n between three
                                              particles down", "real");
87                               }
88                    } else if(vars.type == 'D') {
89                             vars.ku= setup_gen("the attraction between two particles,
                                      horizonatal second hexagon","real");
90                             vars.kd= setup_gen("the attraction between two particles,
                                      vertical second hexagon", "real");
91                    }
92             }
93        // Finding the overal field
94        vars.h= setup_gen("the overall field", "real");
95        // Setting the number of intial values to ignore
96        ignore= (int)setup_gen("how many tens of inital values to ignore", "integer");
97        while(ignore >= MODELS || ignore < 0) {
98                // in case they got it wrong
99                printf("Error: too many values ignored. Please enter a positive number
                          less than %d\n", MODELS);
100               ignore= (int)setup_gen("How many inital values to ignore", "integer");
101       }
102       while(min == max) {
103               //setting the minimum and maximum temperatures
104               min= (float)setup_gen("minimum temperature", "real above zero");
105               max= (float)setup_gen("maximum temperature", "real above zero and min"
                          );
106               if(min < 0) min= 0; //As we are working in Kelvin it makes little
                          sense to have negative numbers
107               if(max < 0) max= 0;
108       }
109       //to save retyping if the temperatures are inputted the wrong way round
110       if(min > max) {
111               minmax= min;
112               min= max;
113               max= minmax;
114       }
115       printf("Enter output filename: ");
116       if(fscanf(stdin, "%s", &filename) != NULL) {
117               printf("\n");
118               fflush(stdin);
119       }
120       timer= time(NULL); // starting the timer
121       // seeding the random numbers with the current time
122       srand((unsigned)time(NULL));
123       // creating our lattice
124       chain= safe_malloc(length * sizeof(int [height]),"Allocating Memory to Chain")
                  ;
125       // The simulation starts here
126       initialise_chain(chain);
```

```
127            coord= hol1;
128            if(vars.type == 'U') {
129                    lattice= safe_malloc(5 * sizeof(int),"Holding array for spin");
130                    split= safe_malloc(5 * sizeof(double),"Holding array for average spin"
                            );
131            } else {
132                    lattice= safe_malloc(3 * sizeof(int),"Holding array for spin");
133                    split= safe_malloc(3 * sizeof(double),"Holding array for average spin"
                            );
134            }
135            for(point= 0; point < array_len; point++) {
136                    // getting a temperature
137                    set_temp(/*coord,*/ min, max);
138                    // starting our average anew
139                    coord= hol1;
140                    lattice[0]= length*height;
141                    if(vars.type == 'T') {
142                            lattice[1]= 2*(((length-1)*(height-1))+((length-1)+(height-1))
                                    );
143                    } else if (vars.type == 'U') {
144                            lattice[1]= length*height*0.5;
145                            lattice[2]= length*height*0.5;
146                            lattice[3]= (((length-1)*(height-1))+((length-1)+(height-1)));
147                            lattice[4]= (((length-1)*(height-1))+((length-1)+(height-1)));
148
149                    }
150                    // setting the chain to "UP"
151                    if(point != 0) initialise_chain(chain);
152                    for(cycle= 0; cycle < MODELS; cycle++) {
153                            for(count=0; count < ITERATIONS; count++) {
154                                    //running the metropolis algorithm
155                                    metropolis(chain, coord->temp, vars, split, lattice);
                                            // Performing the algorithm
156                                    if(cycle == ignore && count == 0) {
157                                            split[0]= (double)lattice[0]/(length*height);
158                                            if(vars.type == 'T') {
159                                                    split[1]= (double)lattice[1]/(2*(((
                                                            length-1)*(height-1))+((length-1)
                                                            +(height-1))));
160                                            } else if (vars.type == 'U') {
161                                                    split[1]= (double)lattice[1]/(length*
                                                            height*0.5);
162                                                    split[2]= (double)lattice[2]/(length*
                                                            height*0.5);
163                                                    split[3]= (double)lattice[3]/(2*(((
                                                            length-1)*(height-1))+((length-1)
                                                            +(height-1))));
164                                                    split[4]= (double)lattice[4]/(2*(((
                                                            length-1)*(height-1))+((length-1)
                                                            +(height-1))));
```

```
165                                                    }
166                                                  }
167                                              }
168                                  percent= ((float)(cycle+1+(MODELS*point))/(float)(MODELS*
                                         array_len));
169                                  seconds= (time(NULL)-timer);
170                                  timeleft= (long int)(((time(NULL)-timer)/percent)-seconds);
171                                  seconds= timeleft%60;
172                                  minutes= abs(timeleft/60)%60;
173                                  hours= abs(timeleft/(60*60));
174                                  printf("%.2f %% done, time left: %.2d hours, %.2d minutes, %.2
                                         d seconds\r",percent*100, hours, minutes,seconds); // To
                                         show the program is still working
175                              }
176                      coord->p= (double)split[0]/(double)((MODELS-ignore)*ITERATIONS); //
                             storing the average spin
177                      if(vars.type == 'T' || vars.type == 'U') {
178                              coord->q= (double)split[1]/(double)((MODELS-ignore)*ITERATIONS
                                 );
179                      }
180                      if(vars.type == 'U') {
181                              coord->r= (double)split[2]/(double)((MODELS-ignore)*ITERATIONS
                                 );
182                              coord->tripu= (double)split[3]/(double)((MODELS-ignore)*
                                 ITERATIONS);
183                              coord->tripd= (double)split[4]/(double)((MODELS-ignore)*
                                 ITERATIONS);
184                      }
185                      printf("\nThe average is %.2f\n",coord->p); // a check to see if it is
                             resonable
186                      printf("The temperature is %.2f\n",coord->temp);
187                      printf("Point: %d\n", point); // where we are up to
188              }
189      write_2d_array (filename, vars); // outputing to our data file
190      free_cond();
191      printf ("Now run maple to plot graph\n"); // handy hint to the user
192      free(chain);
193      free(lattice);
194      free(split);
195      seconds= (time(NULL)-timer)%60;
196      minutes= (abs((time(NULL)-timer-seconds)/60))%60;
197      hours= abs((time(NULL)-timer-seconds-(minutes*60))/(60*60));
198      printf("Runtime: %ld hours, %ld minutes, %ld seconds\n", hours, minutes,
             seconds); // how long it too
199      return 0; // finish
200 }
```

B.2.3 setup2d.c

```
 1  %auto−ignore
 2  /* We link our files with a standard header */
 3  #include "isling2dt.h"
 4
 5  void *safe_malloc (size_t size, char *location) {
 6          /* This function performs a malloc operation but checks to
 7          see if there is enough free memory for this to be written to. */
 8          void *ptr;
 9          ptr= malloc(size);
10          if (ptr == NULL) {
11                  printf("Out of memory at function: %s\n",location);
12                  exit(−1);
13          }
14          return ptr;
15  }
16
17  void *safe_realloc(int *array, size_t size, char *location) {
18          /* This is like the safe_malloc but for realloc. */
19          void *ptr;
20          ptr= realloc(array, size);
21          if (ptr == NULL) {
22                  printf("Out of memory at function: %s\n",location);
23                  exit(−1);
24          }
25          return ptr;
26  }
27
28  int setup_chain(char *quant) {
29          /* This function is a left over from a previous program for
30          the one dimensional Ising model. A two dimensional version
31          is being looked into.*/
32          int num;
33          printf("Enter the %s of chain required: (0,1,...) ",quant);
34          fscanf(stdin, "%d", &num);
35          printf("\n");
36          return num;
37  }
38
39  void initialise_chain(int chain[length][height]) {
40          /* This function take the chain and set all site in the "UP"
41          position */
42          int i,j;
43          for(j=0; j < height; j++){
44                  for(i=0; i < length; i++){
45                          if(chain[i][j] != UP) chain[i][j]= UP;
46                  }
47          }
48  }
```

```
49
50    double setup_gen(char *item, char *units) {
51            /* This is the basic input function, as so many of our variables
52            have similar questions I decided to write a function for them*/
53            double i;
54            printf("Enter the value of the %s: (%s) ",item,units);
55            if (fscanf(stdin, "%lf", &i)) {
56            printf("\n");
57            } else {
58            printf("\n ERROR: No input found. \n");
59            clearerr(stdin);
60            fflush(stdin);
61            setup_gen(item,units);
62            fflush(stdin);
63            }
64            return i;
65    }
66
67    void write_2d_array (char output[], ACT action)
68    /* Take the arrays in x and y both of length array_len and write a file in
69    maple output format to the file named filename */
70    {
71            int i;
72            int m= array_len;
73            COORD *coord;
74            FILE *fptr;
75            coord= hol1;
76            fptr= fopen (output, "w");
77            fprintf (fptr, "p:=[\n");
78            while(coord != NULL) {
79                    fprintf (fptr, "[%.3f,%.15f]",coord->temp,coord->p);
80                    if (/*m > 1*/coord->next != NULL) {
81                        fprintf (fptr, ",");
82                    }
83                    fprintf (fptr, "\n");
84                    m=array_len-1-i;
85                    coord= coord->next;
86            }
87            fprintf (fptr, "];\n");
88            m= array_len;
89            if(action.type == 'T' || action.type == 'U') {
90                    if(action.type == 'T') {
91                            fprintf (fptr, "trip:= [\n");
92                    } else {
93                            fprintf (fptr, "q:=[\n");
94                    }
95                    coord= hol1;
96                    while(coord != NULL) {
97                            fprintf (fptr, "[%.3f,%.12f]",coord->temp,coord->q);
98                            if (/*m > 1*/coord->next != NULL) {
```

```
 99                                    fprintf (fptr, ",");
100                            }
101                            fprintf (fptr, "\n");
102                            m=array_len -1-i;
103                            coord= coord->next;
104                     }
105                 fprintf (fptr, "];\n");
106                 m= array_len;
107                 if(action.type == 'U') {
108                        fprintf (fptr, "r:=[\n");
109                        coord= hol1;
110                        while(coord != NULL) {
111                                fprintf (fptr, "[%.3f,%.12f]",coord->temp,coord->r);
112                                if (/*m > 1*/ coord->next != NULL) {
113                                        fprintf (fptr, ",");
114                                }
115                                fprintf (fptr, "\n");
116                                m=array_len -1-i;
117                                coord= coord->next;
118                        }
119                        fprintf (fptr, "];\n");
120                        m= array_len;
121                        fprintf (fptr, "tripu:=[\n");
122                        coord= hol1;
123                        while(coord != NULL) {
124                                fprintf (fptr, "[%.3f,%.12f]",coord->temp,coord->tripu
                                           );
125                                if (/*m > 1*/coord->next != NULL) {
126                                        fprintf (fptr, ",");
127                                }
128                                fprintf (fptr, "\n");
129                                m=array_len -1-i;
130                                coord= coord->next;
131                        }
132                        fprintf (fptr, "];\n");
133                        m= array_len;
134                        fprintf (fptr, "tripd:=[\n");
135                        coord= hol1;
136                        while(coord != NULL) {
137                                fprintf (fptr, "[%.3f,%.12f]",coord->temp,coord->tripd
                                           );
138                                if (/*m > 1*/coord->next != NULL) {
139                                        fprintf (fptr, ",");
140                                }
141                                fprintf (fptr, "\n");
142                                m=array_len -1-i;
143                                coord= coord->next;
144                        }
145                        fprintf (fptr, "];\n");
146                 }
```

```
147                    }
148                    fprintf (fptr, "J:=[\n");
149                    fprintf (fptr, "%.12f,%.12f,%.12f,%.12f",action.jh, action.jv, action.jh2,
                            action.jv2);
150                    fprintf (fptr, "];\n");
151                    fprintf (fptr, "Jay:= %.12f;", action.jdu);
152                    fprintf (fptr, "Jdash:= %.12f;", action.jdd);
153                    fprintf (fptr, "TripJay:= %.12f;", action.ku);
154                    fprintf (fptr, "TripJdash:= %.12f;", action.kd);
155                    fclose (fptr);
156   }
157
158   void set_temp(/*float temp[4][array_len], int point,*/ float min, float max) {
159                    /* A little function, to calculate the temperature points over the
160                    range between the maximum and minimum temperature levels. The random
161                    number is between 0 and 1000, and then divided by 1000 and multiplied
162                    by the range so that we can get 200 unique points */
163                    int i;
164                    float range, test;
165                    COORD *coord;
166                    range= max-min; //working out the range of the temperatures
167                    test= 0; //so  our loop runs at least once.
168                    //To avoid divide by zero errors
169                    while (test == 0) {
170                            test= (((float)(rand()%1000)/1000) * range)+ min;
171                    }
172                    coord= hol1;
173                    //To guarentee unique temperatures we check the current value against all
                            previous values
174                    while(coord != NULL) {
175                            while (test == 0 || test == coord->temp) {
176                                    test= (((float)(rand()%1000)/1000) * range)+ min;
177                                    coord= hol1;
178                            }
179                            coord= coord->next;
180                    }
181                    coord= safe_malloc(sizeof(COORD),"Setting a temperature point");
182                    coord->temp= test;
183                    coord->next= hol1;
184                    hol1= coord;
185   }
186
187   void free_cond (void) {
188                    /* This frees up the memory taken by the linked list of
189                    conditions by deleting each list item one after the other
190                    from the begining to the end. */
191                    COORD *del_ptr;
192                    while (hol1 != NULL) {
193                            del_ptr= hol1;
194                            hol1= hol1->next;
```

```
195                        free(del_ptr);
196            }
197    }
```

B.2.4   mach2dt.c

```
1   %auto−ignore
2   /* We link our files with a standard header */
3   #include "isling2dt.h"
4
5   void metropolis(int chain[length][height], float temp, ACT var,double split[3],
6     int lattice[3]) {
7           /* This is the same as the previous function but now for the Union Jack
                   lattice */
8           int i= rand()%length;
9           int j= rand()%height;
10          int h, k;
11          double changef= 0;
12          double changeo= 0;
13          double tripo;
14          double tripf;
15          double tripod;
16          double tripfd;
17          double prob;
18          double condition;
19
20
21          changeo= energyb(chain, var, chain[i][j], i, j);
22          changef= energyb(chain, var, −chain[i][j], i, j);
23          if(var.type == 'T') {
24                  tripo= chain[i][j] * ((chain[(i+1)%length][(j+1)%height]*(chain[i][(j
                          +1)%height]
25                  + chain[(i+1)%length][j])) + (chain[(i+length−1)%length][(j+height−1)%
                          height]*
26                  (chain[i][(j+height−1)%height]+chain[(i+length−1)%length][j]))+ (chain
                          [(i+1)%length][j]
27                  * chain[i][(j+height−1)%height])+(chain[i][(j+1)%height]*chain[(i+
                          length−1)%length][j]));
28                  tripf= (−chain[i][j]) * ((chain[(i+1)%length][(j+1)%height]*(chain[i
                          ][(j+1)%height]
29                  + chain[(i+1)%length][j])) + (chain[(i+length−1)%length][(j+height−1)%
                          height]*
30                  (chain[i][(j+height−1)%height]+chain[(i+length−1)%length][j]))+ (chain
                          [(i+1)%length][j]
31                  * chain[i][(j+height−1)%height])+(chain[i][(j+1)%height]*chain[(i+
                          length−1)%length][j]));
32          }else if(var.type == 'U' && (i+j)%2 == 0) {
33                  tripo= chain[i][j]*((chain[(i+1)%length][(j+1)%height]*(chain[(i+1)%
                          length][j]
34                  + chain[i][(j+1)%height]))+ (chain[(i+length−1)%length][(j+height−1)%
                          height]
35                  *(chain[i][(j+height−1)%height]+chain[(i+length−1)%length][j])));
36                  tripod= chain[i][j]*((chain[(i+length−1)%length][(j+1)%height]
37                  *(chain[(i+length−1)%length][j]+chain[i][(j+1)%height]))
```

```
38                  + (chain[(i+1)%length][(j+height-1)%height]*(chain[i][(j+height-1)%
                       height]+chain[(i+1)%length][j])));
39                  tripf= (-chain[i][j])*((chain[(i+1)%length][(j+1)%height]*(chain[(i+1)
                       %length][j]
40                  + chain[i][(j+1)%height]))+ (chain[(i+length-1)%length][(j+height-1)%
                       height]
41                  *(chain[i][(j+height-1)%height]+chain[(i+length-1)%length][j])));
42                  tripfd= (-chain[i][j])*((chain[(i+length-1)%length][(j+1)%height]
43                  *(chain[(i+length-1)%length][j]+chain[i][(j+1)%height]))
44                  + (chain[(i+1)%length][(j+height-1)%height]*(chain[i][(j+height-1)%
                       height]+chain[(i+1)%length][j])));
45          } else if (var.type == 'U' && (i+j)%2 != 0) {
46                  tripo= chain[i][j]*((chain[i][(j+1)%height]*chain[(i+length-1)%length
                       ][j])+(chain[(i+1)%length][j] * chain[i][(j+height-1)%height]));
47                  tripod= chain[i][j]* ((chain[i][(j+1)%height]*chain[(i+1)%length][j])
                       +(chain[(i+length-1)%length][j]*chain[i][(j+height-1)%height]));
48                  tripf= (-chain[i][j])*((chain[i][(j+1)%height]*chain[(i+length-1)%
                       length][j])+(chain[(i+1)%length][j] * chain[i][(j+height-1)%height
                       ]));
49                  tripfd= (-chain[i][j])* ((chain[i][(j+1)%height]*chain[(i+1)%length][j
                       ])+(chain[(i+length-1)%length][j]*chain[i][(j+height-1)%height]));
50
51          }
52      if(0 >= - changeo + changef) {
53              lattice[0]= lattice[0] - 2*chain[i][j];
54              if((i+j)%2 == 0  && var.type == 'U') {
55                      lattice[1]= lattice[1] - 2*chain[i][j];
56              } else if ((i+j)%2 !=0 && var.type == 'U') {
57                      lattice[2]= lattice[2] - 2*chain[i][j];
58              }
59              if(var.type == 'T') {
60                      lattice[1]= lattice[1] - tripo + tripf;
61              } else if (var.type == 'U') {
62                      lattice[3]= lattice[3] - tripo + tripf;
63                      lattice[4]= lattice[4] - tripod + tripfd;
64              }
65              chain[i][j]= -chain[i][j];
66      } else {
67              prob= exp(-(double)(1/(BOLT * temp))*(-changeo + changef));
68              condition= COND*prob;
69              if(rand()%COND < condition) {
70                      lattice[0]= lattice[0] - 2*chain[i][j];
71                      if((i+j)%2 == 0 && var.type == 'U') {
72                              lattice[1]= lattice[1] - 2*chain[i][j];
73                      } else if ((i+j)%2 !=0 && var.type == 'U') {
74                              lattice[2]= lattice[2] - 2*chain[i][j];
75                      }
76                      if(var.type == 'T') {
77                              lattice[1]= lattice[1] - tripo + tripf;
78                      } else if(var.type == 'U') {
```

```
79                                              lattice[3]= lattice[3] − tripo + tripf;
80                                              lattice[4]= lattice[4] − tripod + tripfd;
81                                    }
82                              chain[i][j]= −chain[i][j];
83                        }
84              }
85              split[0]= split[0] + (double)lattice[0]/(length*height);
86              if(var.type == 'T') {
87                        split[1]= split[1] + (double)lattice[1]/(2*(((length−1)*(height−1))+((
                               length−1)+(height−1))));
88              } else if(var.type == 'U') {
89                        split[1]= split[1] + (double)lattice[1]/(0.5*length*height);
90                        split[2]= split[2] + (double)lattice[2]/(0.5*length*height);
91                        split[3]= split[3] + (double)lattice[3]/(((length−1)*(height−1))+((
                               length−1)+(height−1)));
92                        split[4]= split[4] + (double)lattice[4]/(((length−1)*(height−1))+((
                               length−1)+(height−1)));
93              }
94    }
95
96    double energy(int chain[length][height], ACT cond) {
97              /* This is the energy calculation for the triangular and square lattices
98              and returns the resultant value back */
99              int i, j, k, m;
100             double total;
101             double holdu= cond.ku;
102             double holdd= cond.kd;
103             double hu= cond.jdu;
104             double hd= cond.jdd;
105             double hv= cond.jv;
106             double hh= cond.jh;
107             double sum1= 0;
108             double sum2= 0;
109             double sum3= 0;
110             k= 0;
111             for (i=0; i < length; i++) {
112                       if(k == 1) m= 3;
113                       if(k == 0) m= 0;
114                       for (j=0; j < height; j++) {
115                                 if(cond.type == 'U') {
116                                           switch(k) {
117                                                     case 0:
118                                                               holdd= cond.kd;
119                                                               holdu= 0;
120                                                               hd= cond.jdd;
121                                                               hu= 0;
122                                                               hh= cond.jh2;
123                                                               hv= cond.jv2;
124                                                               k= 1;
125                                                               break;
```

```
126                                    case 1:
127                                        holdd= 0;
128                                        holdu= cond.ku;
129                                        hd= 0;
130                                        hu= cond.jdu;
131                                        hh= cond.jh;
132                                        hv= cond.jv;
133                                        k= 0;
134                                        break;
135                                default: // well you never know it could
                                        happen!
136                                        printf("An error has occured in the
                                            calculation\n");
137                                        //return(-1);
138                            }
139                        }
140                        if(cond.type == 'H') {
141                            switch(k) {
142                                case 0:
143                                        hv= cond.jv;
144                                        hu= 0;
145                                        hd= 0;
146                                        hh= 0;
147                                        k= 1;
148                                        break;
149                                case 1:
150                                        hv= cond.jv;
151                                        hu= 0;
152                                        hd= 0;
153                                        hh= cond.jh;
154                                        k= 0;
155                                        break;
156                                default: // well you never know it could
                                        happen!
157                                        printf("An error has occured in the
                                            calculation\n");
158                                        return(-1);
159                            }
160                        }
161                        if(cond.type == 'D') {
162                            holdu= 0;
163                            holdd= 0;
164                            switch(m) {
165                                case 0:
166                                        hv= cond.kd;
167                                        hu= cond.jv;
168                                        hd= 0;
169                                        hh= cond.ku;
170                                        m= 1;
171                                        break;
```

```
172                                    case 1:
173                                            hv= 0;
174                                            hu= cond.kd;
175                                            hd= 0;
176                                            hh= 0;
177                                            m= 0;
178                                            break;
179                                    case 2:
180                                            hv= cond.jv;
181                                            hu= 0;
182                                            hd= 0;
183                                            hh= cond.jh;
184                                            m= 0;
185                                            break;
186                                    case 3:
187                                            hv= cond.kd;
188                                            hu= 0;
189                                            hd= cond.jv;
190                                            hh= 0;
191                                            m= 4;
192                                            break;
193                                    case 4:
194                                            hv= 0;
195                                            hu= 0;
196                                            hd= cond.kd;
197                                            hh= cond.jh;
198                                            m= 5;
199                                            break;
200                                    case 5:
201                                            hv= cond.jv;
202                                            hu= 0;
203                                            hd= 0;
204                                            hh= cond.ku;
205                                            m= 3;
206                                            break;
207                                    default: // well you never know it could
                                            happen!
208                                            printf("An error has occured in the
                                                    calculation\n");
209                                            return(-1);
210                            }
211                    }
212                    // Calculation for nearest neighbours
213                    sum1= sum1 + (chain[i][j]*((hh*chain[(i+1)%length][j])
214                    + (hv*chain[i][(j+1)%height]) + (hu*chain[(i+1)%length][(j+1)%
                            height])))
215                    + (hd*chain[i][(j+1)%height]*chain[(i+1)%length][j]);
216                    // Calculation for triplet neighbours
217                    sum3= sum3 + (holdu*chain[i][j]*(chain[(i+1)%length][(j+1)%
                            height]
```

```
218                                      * ( chain [( i+1)%length ] [ j]+chain [ i ] [( j+1)%height ] ) ) )
219                                      + ( holdd*chain [ i ] [( j+1)%height ]*( chain [( i+1)%length ] [ j ]
220                                      * ( chain [( i+1)%length ] [( j+1)%height]+chain [ i ] [ j ]) ) ) ; ;
221                                      // Calculation for sites
222                                      sum2= sum2 + chain [ i ] [ j ] ;
223                              }
224                      if ( i%2 == 0) { // to allow the intial diagonal of the row to alternate
225                              k= 1;
226                      } else {
227                              k= 0;
228                      }
229              }
230      // adding the three sums together and applying required factors
231      total= (−sum3)− sum1 − ( cond . h*sum2 ) ;
232      // returning the result
233      return total ;
234 }
235
236
237 double energyb ( int chain [ length ] [ height ] , ACT cond , int change , int i , int j ) {
238      /* This is the energy calculation for the triangular and square lattices
239      and returns the resultant value back */
240      int k , m;
241      double total ;
242      double holdu= cond . ku ;
243      double holdd= cond . kd ;
244      double holdua= holdu ;
245      double holdda= holdd ;
246      double hu= cond . jdu ;
247      double hd= cond . jdd ;
248      double hv= cond . jv ;
249      double hh= cond . jh ;
250      double hua= hu ;
251      double hva= cond . jv2 ;
252      double hda= hd ;
253      double hha= cond . jh2 ;
254      double sum1= 0;
255      double sum2= 0;
256      double sum3= 0;
257      if (( i+j )%2 == 0) {
258              k= 1;
259      } else {
260              k= 0;
261      }
262      if ( i%2 == 0 && cond . type == 'D' ) {
263              switch ( j%3) {
264                      case 0:
265                              m= 0;
266                              break ;
267                      case 1:
```

```
268                                         m= 1;
269                                     break;
270                         case 2:
271                                     m= 2;
272                                     break;
273                         default:
274                                     printf("An error has occured");
275                                     exit(-1);
276                 }
277         } else if(cond.type == 'D') {
278                 switch(j%3) {
279                         case 0:
280                                     m=3;
281                                     break;
282                         case 1:
283                                     m= 4;
284                                     break;
285                         case 2:
286                                     m= 5;
287                                     break;
288                         default:
289                                     printf("An error has occured");
290                                     exit(-1);
291                 }
292         }
293
294         if(cond.type == 'U') {
295                 switch(k) {
296                         case 0:
297                                     holdd= 0;
298                                     holdda= cond.kd;
299                                     holdu= 0;
300                                     holdua= cond.ku;
301                                     hd= 0;
302                                     hda= 0;
303                                     hu= 0;
304                                     hua= 0;
305                                     hh= cond.jh2;
306                                     hv= cond.jv2;
307                                     hha= cond.jh;
308                                     hva= cond.jv;
309                                     k= 1;
310                                     break;
311                         case 1:
312                                     holdd= cond.kd;
313                                     holdda= 0;
314                                     holdu= cond.ku;
315                                     holdua= 0;
316                                     hd= cond.jdd;
317                                     hda= cond.jdd;
```

```
318                                    hu= cond.jdu;
319                                    hua= cond.jdu;
320                                    hh= cond.jh;
321                                    hv= cond.jv;
322                                    hha= cond.jh2;
323                                    hva= cond.jv2;
324                                    k= 0;
325                                    break;
326                          default: // well you never know it could happen!
327                                    printf("An error has occured in the calculation \n");
328                                    //return(-1);
329                    }
330            }
331        if(cond.type == 'H') {
332                holdd= 0;
333                holdda= 0;
334                holdu= 0;
335                holdua= 0;
336                switch(k) {
337                      case 0:
338                                    hu= 0;
339                                    hua= 0;
340                                    hd= 0;
341                                    hda= 0;
342                                    hh= 0;
343                                    hha= cond.jh;
344                                    k= 1;
345                                    break;
346                      case 1:
347                                    hu= 0;
348                                    hua= 0;
349                                    hd= 0;
350                                    hda= 0;
351                                    hh= cond.jh;
352                                    hha= 0;
353                                    k= 0;
354                                    break;
355                          default: // well you never know it could happen!
356                                    printf("An error has occured in the calculation \n");
357                                    return(-1);
358                    }
359            }
360        if(cond.type == 'D') {
361                holdu= 0;
362                holdua= 0;
363                holdd= 0;
364                holdda= 0;
365                switch(m) {
366                      case 0:
367                                    hv= cond.kd;
```

```
368                             hva= cond.kd;
369                             hu= cond.jv;
370                             hua= cond.jv;
371                             hd= 0;
372                             hh= cond.ku;
373                             hha= cond.jh;
374                             m= 1;
375                             break;
376                     case 1:
377                             hv= 0;
378                             hu= cond.kd;
379                             hua= cond.kd;
380                             hd= 0;
381                             hh= 0;
382                             hha= cond.ku;
383                             m= 0;
384                             break;
385                     case 2:
386                             hv= cond.jv;
387                             hva= cond.jv;
388                             hu= 0;
389                             hd= 0;
390                             hh= cond.jh;
391                             m= 0;
392                             break;
393                     case 3:
394                             hv= cond.kd;
395                             hva= cond.kd;
396                             hu= 0;
397                             hd= 0;
398                             hh= 0;
399                             hha= cond.ku;
400                             m= 4;
401                             break;
402                     case 4:
403                             hv= 0;
404                             hu= 0;
405                             hd= cond.kd;
406                             hda= cond.kd;
407                             hh= cond.jh;
408                             m= 5;
409                             break;
410                     case 5:
411                             hv= cond.jv;
412                             hva= cond.jv;
413                             hu= 0;
414                             hd= cond.kd;
415                             hda= cond.kd;
416                             hh= cond.ku;
417                             hha= cond.jh;
```

```
418                              m= 3;
419                                break;
420                    default: // well you never know it could happen!
421                          printf("An error has occured in the calculation\n");
422                          return(-1);
423              }
424        }
425        // Calculation for nearest neighbours
426        sum1= sum1 + (change*((hh*(chain[(i+1)%length][j])
427        + (hv*chain[i][(j+1)%height]) + (hu*chain[(i+1)%length][(j+1)%height])
428        + (hd*chain[(i+1)%length][(j+(height-1))%height])
429        + (hha*chain[(i+(length-1))%length][j]) + (hva*chain[i][(j+(height-1))%height
                ])
430        + (hua*chain[(i+(length-1))%length][(j+height-1)%height])
431        + (hda*chain[(i+(length-1))%length][(j+1)%height]))));
432        // Calculation for triplet neighbours
433        sum3= sum3 + (change*((holdu*chain[(i+1)%length][(j+1)%height]
434        * (chain[(i+1)%length][j]+chain[i][(j+1)%height]))
435        + (holdu*chain[(i+length-1)%length][(j+height-1)%height]
436        * (chain[(i+length-1)%length][j]+chain[i][(j+height-1)%height]))
437        + (holdd*chain[(i+1)%length][(j+height-1)%height]
438        * (chain[i][(j+height-1)%height]+chain[(i+1)%length][j]))
439        + (holdd*chain[(i+length-1)%length][(j+1)%height]
440        * (chain[i][(j+1)%height]+chain[(i+length-1)%length][j]))
441        + (holdda*chain[(i+1)%length][j]*chain[i][(j+1)%height])
442        + (holdua*chain[(i+1)%length][j]*chain[i][(j+height-1)%height])
443        + (holdda*chain[(i+length-1)%length][j]*chain[i][(j+height-1)%height])
444        + (holdua*chain[(i+length-1)%length][j]*chain[i][(j+1)%height])));
445        // Calculation for sites
446        sum2= sum2 + change;
447        // adding the three sums together and applying required factors
448        total= (-sum3)- sum1 - (cond.h*sum2);
449        // returning the result
450        return total;
451 }
```

Programming II
Mean Field Approximation

C.1 Programmers guide

The broad purpose of this code is to perform mean field simulations of the Ising model on various lattices. As with the last program, it does this by asking the user initially for the conditions for the system that is to be studied. These include the shape of the lattice, the temperature range to be studied over and the interaction strengths of the lattice. It then performs simulations of systems at temperatures at regular intervals along the temperature range. The program then outputs the results to a file which can be read into a mathematical program.

C.1.1 Flow

Initially the user is asked for the initial conditions of the system they wish to study, and then the temperature range they would like to study. After these have been stored, the user is then asked how many data points they require. After this the program calculates each data point by iteratively using the formulas from Chapter 5 until either the value of the sublattice magnetisation is constant, or 10,000 iterations have taken place, whichever occurs first. The value of magnetisation is then recorded for both the partially uncoupled and coupled equations along with the temperature. The program then moves on to the next data point, using the previous magnetisation value as the initial value. The temperature values are calculated at regular intervals in the temperature range. After the required number of data points have been calculated, the

results are output to a data file which can then be read into a mathematical program. Again the simulation time is given on the screen after the program has finished.

C.2 Code listings

C.2.1 mean1.h

```
1  %auto−ignore
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <math.h>
5  #include <time.h>
6
7  #define BOLT 8.617343*pow(10,−5)     /*The boltzman contant*/
8
9  typedef struct coordinate {
10         long double sigma;
11         long double mu;
12         long double unsigma;
13         long double unmu;
14         float temp;
15         struct coordinate *next;
16  } COORD;
17
18  double setup_gen(char *, char *);
19  void *safe_malloc(size_t, char *);
20  void *safe_realloc (int *, size_t, char *); /* Does the same for realloc */
21  void write_2d_array (char *, double, double, double, char);
22  void free_cond (void);            /* Frees the memory taken by linked list */
23  long double ltanh(long double);
24
25  extern COORD *hol1;
```

C.2.2   mean1.c

```
1   %auto−ignore
2   #include "mean1.h"
3
4   COORD ∗hol1= NULL;
5
6   int main() {
7           int i, timeleft, points, seconds, minutes, hours, c, d;
8           time_t timer;
9           COORD ∗coord;
10          float min, max, minmax, percent, interval, T, q;
11          char type;
12          double J, K, B;
13          long double x, y, u, v, beta, m, n, a, b;
14          char filename[100];
15
16          x= 0;
17          y= 0;
18          while(type != 'S' && type != 'T' && type != 'U' && type != 'N' && type != 'H'
                  && type != 'D') {
19                  printf("What type of lattice would you like to study? \n ([S]quare, [T
                          ]riangular, [U]nion Jack, [N]ext Nearest, [H]exagonal, [D]ouble
                          Hexagonal) ");
20                  fscanf(stdin, "%1s", &type);
21                  type= toupper(type);
22          }
23          switch(type) {
24                  case 'S':
25                          q= 2;
26                          break;
27                  case 'T':
28                          q= 3;
29                          break;
30                  case 'U':
31                          q= 2;
32                          break;
33                  case 'H':
34                          q=1.5;
35                          break;
36                  default :
37                          printf("There has been an error");
38                          exit(−1);
39          }
40          if(type == 'U') {
41                  J= setup_gen("interaction of the square lattice", "real");
42                  K= setup_gen("interaction of the diagonal lattice", "real");
43          } else {
44                  J= setup_gen("interaction of nearest neighbours", "real");
45          }
```

```
46              B= setup_gen("strength of overal field", "real");
47              x= (long double)setup_gen("intial value for sigma", "real, less than or equal
                    to +/- 1");
48          if(type == 'U') {
49                  y= (long double)setup_gen("intial value for mu", "real, less than or
                        equal to +/- 1");
50          }
51          min= 0;
52          max= min;
53          while(min == max) {
54                  min= (float)setup_gen("minimum temperature", "real above zero");
55                  max= (float)setup_gen("maximum temperature", "real above zero");
56                  if(min < 0) min= 0; //As we are working in Kelvin it makes little
                            sense to have negative numbers
57                  if(max < 0) max= 0;
58          }
59          //to save retyping if the temperatures are inputted the wrong way round
60          if(min > max) {
61                  minmax= min;
62                  min= max;
63                  max= minmax;
64          }
65          points= (int)setup_gen("data points", "integer above zero");
66          printf("Enter output filename: ");
67          if(fscanf(stdin, "%s", &filename) != NULL) {
68                  printf("\n");
69                  fflush(stdin);
70          }
71          timer= time(NULL); // starting the timer
72          // creating our temperature array
73          interval= (max-min)/points;
74          c= 0;
75          d= 0;
76          coord= hol1;
77          coord= safe_malloc(sizeof(COORD), "New data point");
78          coord->temp= min;
79          coord->sigma= x;
80          coord->mu= y;
81          coord->unsigma= x;
82          coord->unmu= y;
83          coord->next= hol1;
84          hol1= coord;
85          u= x;
86          v= y;
87          m= 10;
88          n= 10;
89          a= 10;
90          b= 10;
91          for(i= 1; i < points; i++) {
92                  T= min+(i*interval);
```

```
93                      beta= (long double)(1/(BOLT * T));
94                      while(x != m || y != n) {
95                              m= x;
96                              n= y;
97                              if(type == 'U') {
98                                      x= beta*((2*K*m)+(2*J*n)+B);
99                                      y= beta*((2*J*m)+B);
100                             } else {
101                                     x= beta*((q*J*m)+B);
102                             }
103                             if(c == 0) {
104                                     x= ltanh(x);
105                                     y= ltanh(y);
106                             } else {
107                                x= (ltanh(x)+ m)/2;
108                                y= (ltanh(y)+ n)/2;
109                             }
110                             c++;
111                             if(c == pow(10,8)) break;
112                      }
113                      c= 0;
114                      while(u != a || v != b && type == 'U') {
115                              a= u;
116                              b= v;
117                              if(d== 0) {
118                                      u= ltanh(beta*((2*K*a)+B));
119                                      v= ltanh(beta*(4*J*a)+B);
120                             } else {
121                                u= (ltanh(beta*((2*K*a)+B))+a)/2;
122                                v= (ltanh(beta*((4*J*a)+B))+b)/2;
123                             }
124                             d++;
125                             if(d == pow(10,8)) break;
126                      }
127                      d= 0;
128                      coord= safe_malloc(sizeof(COORD), "New data point");
129                      coord->temp= T;
130                      coord->sigma= x;
131                      coord->mu= y;
132                      coord->unsigma= u;
133                      coord->unmu= v;
134                      coord->next= hol1;
135                      hol1= coord;
136                      m= 10;
137                      n= 10;
138                      a= 10;
139                      b= 10;
140                      percent= (float) (i+1)/points;
141                      seconds= (time(NULL)-timer);
142                      timeleft= (long int)(((time(NULL)-timer)/percent)-seconds);
```

```
143                      seconds= timeleft%60;
144                      minutes= abs(timeleft/60)%60;
145                      hours= abs(timeleft/(60*60));
146                      printf("%.2f %% done, time left: %.2d hours, %.2d minutes, %.2d seconds
                              \r",percent*100, hours, minutes,seconds); // To show the program is
                              still working
147              }
148         write_2d_array(filename, J, K, B, type);
149         free_cond();
150         printf ("\n Now run maple to plot graph\n"); // handy hint to the user
151         seconds= (time(NULL)-timer)%60;
152         minutes= (abs((time(NULL)-timer-seconds)/60))%60;
153         hours= abs((time(NULL)-timer-seconds-(minutes*60))/(60*60));
154         printf("Runtime: %ld hours, %ld minutes, %ld seconds\n", hours, minutes,
                 seconds); // how long it too
155         return 0;
156 }
157
158 long double ltanh(long double x) {
159         if (fabsl(x) < powl(2.0,(-55.0))) {
160             x= x;
161         } else {
162             x= tanhl(x);
163         }
164         return x;
165 }
```

C.2.3 setup.c

```
1   %auto−ignore
2   /* We link our files with a standard header */
3   #include "isling2dt.h"
4
5   void *safe_malloc (size_t size, char *location) {
6           /* This function performs a malloc operation but checks to
7           see if there is enough free memory for this to be written to. */
8           void *ptr;
9           ptr= malloc(size);
10          if (ptr == NULL) {
11                  printf("Out of memory at function: %s\n",location);
12                  exit(−1);
13          }
14          return ptr;
15  }
16
17  void *safe_realloc(int *array, size_t size, char *location) {
18          /* This is like the safe_malloc but for realloc. */
19          void *ptr;
20          ptr= realloc(array, size);
21          if (ptr == NULL) {
22                  printf("Out of memory at function: %s\n",location);
23                  exit(−1);
24          }
25          return ptr;
26  }
27
28  int setup_chain(char *quant) {
29          /* This function is a left over from a previous program for
30          the one dimensional Ising model. A two dimensional version
31          is being looked into.*/
32          int num;
33          printf("Enter the %s of chain required: (0,1,...) ",quant);
34          fscanf(stdin, "%d", &num);
35          printf("\n");
36          return num;
37  }
38
39  void initialise_chain(int chain[length][height]) {
40          /* This function take the chain and set all site in the "UP"
41          position */
42          int i,j;
43          for(j=0; j < height; j++){
44                  for(i=0; i < length; i++){
45                          if(chain[i][j] != UP) chain[i][j]= UP;
46                  }
47          }
48  }
```

```
49
50  double setup_gen(char *item, char *units) {
51          /* This is the basic input function, as so many of our variables
52          have similar questions I decided to write a function for them*/
53          double i;
54          printf("Enter the value of the %s: (%s) ",item,units);
55          if (fscanf(stdin, "%lf", &i)) {
56          printf("\n");
57          } else {
58          printf("\n ERROR: No input found. \n");
59          clearerr(stdin);
60          fflush(stdin);
61          setup_gen(item,units);
62          fflush(stdin);
63          }
64          return i;
65  }
66
67  void write_2d_array (char output[], ACT action)
68  /* Take the arrays in x and y both of length array_len and write a file in
69  maple output format to the file named filename */
70  {
71          int i;
72          int m= array_len;
73          COORD *coord;
74          FILE *fptr;
75          coord= hol1;
76          fptr= fopen (output, "w");
77          fprintf (fptr, "p:=[\n");
78          while(coord != NULL) {
79                  fprintf (fptr, "[%.3f,%.15f]",coord->temp,coord->p);
80                  if (/*m > 1*/coord->next != NULL) {
81                      fprintf (fptr, ",");
82                  }
83                  fprintf (fptr, "\n");
84                  m=array_len-1-i;
85                  coord= coord->next;
86          }
87          fprintf (fptr, "];\n");
88          m= array_len;
89          if(action.type == 'T' || action.type == 'U') {
90                  if(action.type == 'T') {
91                          fprintf (fptr, "trip:= [\n");
92                  } else {
93                          fprintf (fptr, "q:=[\n");
94                  }
95                  coord= hol1;
96                  while(coord != NULL) {
97                          fprintf (fptr, "[%.3f,%.12f]",coord->temp,coord->q);
98                          if (/*m > 1*/coord->next != NULL) {
```

```
 99                              fprintf (fptr, ",");
100                         }
101                    fprintf (fptr, "\n");
102                    m=array_len −1−i;
103                    coord= coord−>next;
104              }
105         fprintf (fptr, "];\n");
106         m= array_len;
107         if(action.type == 'U') {
108              fprintf (fptr, "r:=[\n");
109              coord= hol1;
110              while(coord != NULL) {
111                    fprintf (fptr, "[%.3f,%.12f]",coord−>temp,coord−>r);
112                    if (/*m > 1*/ coord−>next != NULL) {
113                         fprintf (fptr, ",");
114                    }
115                    fprintf (fptr, "\n");
116                    m=array_len −1−i;
117                    coord= coord−>next;
118              }
119              fprintf (fptr, "];\n");
120              m= array_len;
121              fprintf (fptr, "tripu:=[\n");
122              coord= hol1;
123              while(coord != NULL) {
124                    fprintf (fptr, "[%.3f,%.12f]",coord−>temp,coord−>tripu
                              );
125                    if (/*m > 1*/coord−>next != NULL) {
126                         fprintf (fptr, ",");
127                    }
128                    fprintf (fptr, "\n");
129                    m=array_len −1−i;
130                    coord= coord−>next;
131              }
132              fprintf (fptr, "];\n");
133              m= array_len;
134              fprintf (fptr, "tripd:=[\n");
135              coord= hol1;
136              while(coord != NULL) {
137                    fprintf (fptr, "[%.3f,%.12f]",coord−>temp,coord−>tripd
                              );
138                    if (/*m > 1*/coord−>next != NULL) {
139                         fprintf (fptr, ",");
140                    }
141                    fprintf (fptr, "\n");
142                    m=array_len −1−i;
143                    coord= coord−>next;
144              }
145              fprintf (fptr, "];\n");
146         }
```

```
147                }
148                fprintf (fptr, "J:=[\n");
149                fprintf (fptr, "%.12f,%.12f,%.12f,%.12f",action.jh, action.jv, action.jh2,
                       action.jv2);
150                fprintf (fptr, "];\n");
151                fprintf (fptr, "Jay:= %.12f;", action.jdu);
152                fprintf (fptr, "Jdash:= %.12f;", action.jdd);
153                fprintf (fptr, "TripJay:= %.12f;", action.ku);
154                fprintf (fptr, "TripJdash:= %.12f;", action.kd);
155                fclose (fptr);
156     }
157
158     void set_temp (/*float temp[4][array_len], int point,*/ float min, float max) {
159                /* A little function, to calculate the temperature points over the
160                range between the maximum and minimum temperature levels. The random
161                number is between 0 and 1000, and then divided by 1000 and multiplied
162                by the range so that we can get 200 unique points */
163                int i;
164                float range, test;
165                COORD *coord;
166                range= max-min; //working out the range of the temperatures
167                test= 0; //so  our loop runs at least once.
168                //To avoid divide by zero errors
169                while (test == 0) {
170                       test= (((float)(rand()%1000)/1000) * range)+ min;
171                }
172                coord= hol1;
173                //To guarentee unique temperatures we check the current value against all
                       previous values
174                while(coord != NULL) {
175                       while (test == 0 || test == coord->temp) {
176                              test= (((float)(rand()%1000)/1000) * range)+ min;
177                              coord= hol1;
178                       }
179                       coord= coord->next;
180                }
181                coord= safe_malloc(sizeof(COORD),"Setting a temperature point");
182                coord->temp= test;
183                coord->next= hol1;
184                hol1= coord;
185     }
186
187     void free_cond (void) {
188                /* This frees up the memory taken by the linked list of
189                conditions by deleting each list item one after the other
190                from the begining to the end. */
191                COORD *del_ptr;
192                while (hol1 != NULL) {
193                       del_ptr= hol1;
194                       hol1= hol1->next;
```

```
195                free ( del_ptr );
196        }
197  }
```